Under consideration for publication in Knowledge and Information Systems

Revisiting the Effect of History on Learning Performance: The Problem of the Demanding Lord

George Giannakopoulos[1], Themis Palpanas[2]

[1]Institute of Informatics and Telecommunications, NCSR Demokritos, Greece, e-mail: ggianna@iit.demokritos.gr;
[2] DISI, University of Trento, Italy, e-mail: themis@disi.unitn.eu

Abstract. In a variety of settings ranging from recommendation systems to information filtering, approaches which take into account feedback have been introduced to improve services and user experience. However, as also indicated in the machine learning literature, there exist several settings where the requirements and target concept of a learning system changes over time, which consists a case of "concept drift". In several systems a sliding window over the training instances has been used to follow drifting concepts. However, no general analytic study has been performed on the relation between the size of the sliding window and the average performance of a learning system, since previous works have focused on instantaneous performance and specific underlying learners and data characteristics.

This work proposes an analytic model that describes the effect of memory window size on the prediction performance of a learning system that is based on iterative feedback. The analysis considers target concepts changing over time, either periodically or randomly, using a formulation termed "the problem of the demanding lord". Using a signal-to-noise approach to sketch learning ability of underlying machine learning algorithms, we estimate the average performance of a learning system regardless of its underlying algorithm, and, as a corollary, propose a stepping stone towards finding the memory window that maximizes the average performance for a given drift setting and a specific modeling system. We experimentally support the proposed methodology with very promising results on three synthetic and four real datasets, using a variety of learning algorithms including Support Vector Machines, Naive Bayes, Nearest Neighbor and Decision Trees on classification and regression tasks. The results validate the analysis and indicate very good estimation performance in different settings.

Keywords: concept drift, adaptive learning, demanding lord problem, user modeling, incremental learning

Received Aug 13, 2011
Revised Jul 30, 2012
Accepted Sep 01, 2012

1. Introduction

In the literature it has been argued (Webb, Pazzani and Billsus, 2001) that machine learning methods are not enough to follow changing concepts. This opinion has been otherwise expressed as the inability of classifiers to detect changing context (e.g., a user's preferences), which in turn causes changes to an observed behavior (or concept) a learning algorithm is trying to learn. This implied change of context, its effects on classification and the reactions that tackle the change have been studied in the machine learning and data stream mining communities as the problem of "concept drift". Drifting concepts also appear in a variety of other settings in the real world, e.g., the state of a free market or the traits of the most viewed movie. Even though there have been a number of methodologies to either track or react to concept drift, there has been little *analytic* work on the connection between (the quality of) learning and the learning window. Most existing works rely on experimental results and heuristic rules to determine the window parameter and optimize the learners' performance.

In contrast, we propose[1] a systematic approach that allows us to estimate the average performance of learning algorithms for a range of learning window sizes and concept drift frequencies, within a learning task in the presence of concept drift. This approach will allow practitioners to optimize the average performance of incremental learning systems; moreover, it provides the basis for further analytic study of the connection between the average performance of an incremental learning system and the noise in the training set. The questions we answer with this study are the following.

- How can we model the expected performance of learning algorithms based on knowledge of the characteristics of the abrupt concept drift (also termed "concept shift"), such as the period of occurrence of these drifts?
- How can we estimate the performance of a learner for different window sizes and concept change frequency, regardless of the underlying learning algorithm?

The case of concept shift is very common is cases such as spam filtering, user modeling and real-world data prediction. In spam filtering (e.g., see (Liu and Wang, 2011)), spammers may alter the mechanics of e-mail generation to avoid detection. This change may be instantaneous and often times, radical. Another case of shift in spam e-mail is when a filter trained on the preferences of a user is applied on the mailbox of another user (Fdez-Riverola, Iglesias, Díaz, Méndez and Corchado, 2007). In user modeling, two different users may use a single network address to navigate the Web. A server-side modeling system should be able to adapt to the sudden change of user interests implied, e.g., for a case of web page prefetching. Finally, when trying to predict real-word data, as in market trend prediction, real-life events (e.g., terrorist act, announcement of international co-operation, tax/loan rate change, etc) can significantly and radically impact the distribution of observations.

[1] A preliminary version of this work can be found in (Giannakopoulos and Palpanas, 2010b).

We focus on the functional relation between the window size and the average performance of a learning system in the presence of concept shifts (i.e., abrupt change). We formulate this problem under the label of "the problem of the demanding lord", to facilitate understanding. Then, we propose a generic methodology that allows to a-priori estimate a function of the window size that maximizes average performance of a learning system in the presence of periodic target concept shifts. To realize this methodology we show that one can approximately express the average performance of a system as a function of signal-to-noise ratio, referring to the training instances in the memory window. Then, for a given period of abrupt concept drift (also termed "interest shift") with a given implied context (e.g., user), we analytically express the signal-to-noise ratio at each moment in time as a function of the selected learning window size. We then illustrate that the methodology is also effective when applied on real-world cases, where not all theoretic prerequisites are met.

We stress that we do not propose a new learner for concept drift, neither do we evaluate the performance of learners in the presence of concept drift. Instead, we propose for the first time a theory for modeling the *expected performance* of existing learners in the presence of concept drift, applicable to window optimization in partial memory approaches. We chose the window-based approaches as the basis of this theory, because the context is basic and easily transferrable to more complex problems, as part of our future work.

In summary, we make the following contributions.

- We propose a formulation and analytic solution for the problem of estimating the average performance in incremental learning systems for a given context (e.g., user) in the presence of abrupt concept drift (interest shift).
- We describe a methodology to approximately estimate the performance (accuracy) of a learning algorithm as a function of signal-to-noise in the training set, regardless of the algorithm idiosyncrasies.
- We experimentally validate the proposed approach, using a wide variety of learning algorithms for classification and regression tasks, on both synthetic and real datasets.

In the following sections, we present the related work (Section 2) on concept drift and we formulate the problem we face within this work (Section 3). We then describe the proposed analytic methodology that describes the average performance of a learning algorithm as a function of its memory window size. Then, we experimentally validate the analysis (Sections 5, 6) and finally, we conclude with a discussion of our findings in Section 7.

2. Related Work

There have been several studies with different assumptions on the speed or type of drift. The drift can be slow (a drift) or instant (a shift) (Tsymbal, 2004). It can also be caused by a real change of context (real drift) or by the change on the distribution of arriving instances of the — otherwise fixed distribution — target class (virtual drift). Drift has also appeared as a function of time (Widmer and Kubat, 1996), an previous work describes drift which occurs every 50, 100 and 200 instances as moderately quick, slow and very slow, respectively (Stanley, 2001). The systems and learning algorithms then react in different

ways to the drift, sometimes using fixed window sizes, adaptive window sizes or simply non-window-based methods. Algorithms that use windows either exploit all the available information (full-memory) or only the last given information (no-memory) (Maloof and Michalski, 1999). All the other cases, where a window size between one and infinity is chosen, are termed partial-memory approaches.

In an early work (Schlimmer and Granger, 1986), the problem of "concept attainment" in the presence of noise was indicated and studied in the STAGGER system. The reaction to concept drift was a backtracking methodology that allows changing the current description of the target concept to account for the drift. From that time on a multitude of systems have appeared facing the problem of change in a target learned concept, many of them in the incremental learning domain, which has been studied since the late 80's (Angluin, 1988; Reinke and Michalski, 1988).

In later works, we find approaches where either hard-coded thresholds are used (Mitchell, Caruana, Freitag, McDermott and Zabowski, 1994), or the window is adjusted whenever a shift is detected (Crabtree and Soltysiak, 1998). In order to deal with the drift, in (Widmer and Kubat, 1996), a heuristic algorithm is described (Window Adjustment Heuristic) that adjusts window size, but the authors state that the algorithm requires optimization, as its performance affects the whole system strongly. A heuristic approach to deal with concept drift is also described in (Klinkenberg and Renz, 1998), where we can also find a study of fixed and adaptive windows. However, the study is "interested in finding the best window size for a certain transition period containing the onset of the concept shift", while we look for an overall best window as related to average performance at the limit of time. In this work we also find the distinction between "full memory" and "no memory" learner, which are used to compare against the adaptive window methodology. Gradual forgetting (Koychev and Schwab, 2000) has also been used.

A more recent approach uses small sequences (batches) of statistically significant size to estimate the performance of the classifier over running data (Koychev and Lothian, 2005). When concept drift is detected, it is handled through a window-size optimization procedure (Kiefer, 1953): the process considers the performance of the system to be a unimodal function of the window size and through a divide-and-conquer algorithm searches for the optimal window size.

An alternative study (Kuncheva and Žliobaitė, 2009) approximates the classifier error rate after an instantaneous concept shift to estimate a locally optimal learning window. The classifier error is estimated as a function of the training window, based on a set of assumptions: the classifier is expected to be parametric; the theoretic error rate of the classifier is given; classes are equiprobable; the stream of data is i.i.d. In our work, we require no prior knowledge of the theoretic learner details or theoretic data distribution. We take into account the learner as function between input (training) noise levels and expected performance, which allows us to experimentally estimate the classifier behavior in the presence of noise for a given dataset (see Characteristic Transfer Function in Section 4.1). We overcome the requirement for parametric classifiers, and derive generic formulas applicable to any distribution of data. Overall, we develop a new, theoretical basis connecting the *average performance* of a system (at infinity) to a fixed learning window size, in the presence of concept shift.

Recently, researchers have used "local windows" in sub-parts of models, as in (Núñez, Fidalgo and Morales, 2007) where an incremental decision tree uses local sub-concept adaptive window sizes. The approach performs well in handling

different concept drift speeds, as well as being relatively insensitive to noise and virtual drift. The notion of local optimization is also applied in the SAIRT (Fidalgo-Merino and Nunez, 2011) and the FIRT-DD methods (Ikonomovska, Gama, Sebastião and Gjorgjevik, 2009) to face regression problems under drift. Another approach uses multiple competing windows of different sizes (Lazarescu, Venkatesh and Bui, 2004), that try to tackle the problem of differentiating noise from virtual drift from actual concept drift. The idea is that there are three windows — small, medium, large — each limited to a different size range. Each window maps to a level of persistence for any concept drift apparent in incoming data, so an algorithm can use the best window based on each window's current performance.

Other approaches use a window differently or not at all. In (Lam, Mukhopadhyay, Mostafa and Palakal, 1996), a two-level probabilistic learning approach is used: one level for shift detection, one level for user model learning. Shifts are detected per category of interest. A shift in the interestingness of a category — detected by the analysis of a window of recent feedback — causes redistribution of interest probabilities over categories. In (Widyantoro, Ioerger and Yen, 1999) we find a distinction between short- and long-term changes in user interests for information filtering in a feedback-based system. However, no window is used in this case. In (Maloof and Michalski, 1999), the proposed system learns, not based on the most recent instances, but based on extreme examples and batch learning. Related work (Maloof and Michalski, 2004) applies the strategy proposed in (Maloof and Michalski, 1999) as an extension of the AQ11 and GEM online learning algorithms and offers their "partial memory" counterparts. A boosting-based, alternative approach (Scholz and Klinkenberg, 2007) uses a set of base classifiers, using only the last batches of instances to determine dynamically the training instances per iteration, based on the performance of the classifiers on the current batch of data. In (Stanley, 2001) a Concept Drift Committee (CDC) of decision trees vote for the current classification of instances and each decision tree classifier remains used until its voting record efficiency is reduced below a given threshold. At that point, a new decision tree replaces the retiring classifier. This new classifier is trained only from new concepts. The CVFDT algorithm (Hulten, Spencer and Domingos, 2001) builds upon the VFDT decision tree learner (Domingos and Hulten, 2000), updating statistics of subtrees based on a recent window of examples. When needed, new subtrees appear in the existing tree and replace the old ones that are no longer accurate. The window size is static, but the authors note that it would be important to dynamically adjust their size. In another approach (Zhang, Gao, Zhu and Guo, 2011), a Lazy-Tree (L-Tree) representation is used to keep recent instances for a classification problem. The number of kept instances per iteration is related to the maximum allowed instances per tree node, which is a parameter. If the number of instances in a node of the tree exceeds the maximum, then the oldest node is deleted. In (Ramon, Driessens and Croonenborghs, 2007), a set of tree operators (e.g., split/prune leaf, revise node, prune subtree) are used to update the incremental, relational, first-order decision tree model. The updates occur when a splitting criterion (described in a tree node) is redundant or insufficient to desribe the current context.

A focused study of the mistake rate of a learning algorithm that updates its estimate based on the most recent examples (Kuh, Petsche and Rivest, 1990) identifies bounds for this rate, based on the number of recent examples. In this work, the adaptation to concept drift over time is termed "incremental tracking".

In (Helmbold and Long, 1994), the authors study the problem of tracking a subset of a domain (called the target) which changes gradually over time, under the assumption that the drift occurs slowly. The work connects the VC-dimension (d) of the class of possible targets to the difficulty of attaining the target concept, finally indicating the sampling rate that makes a concept drift trackable.

In this work, we provide, through an analytic framework, a methodology that provides the basis for efficient optimization of window sizes in online learning. A major contribution, other than the analysis itself, is based on the proposal of a signal-to-noise function (see Section 4.1) describing the connection between noisy input and the performance of a learning algorithm. The estimation of this function allows one to optimize the window size without explicit knowledge of the learning algorithm, based on an estimation step that captures the behavior of any learning algorithm in the presence of noise.

The methodology presented in the next paragraphs holds for both classification tasks (e.g., classifying incoming mail (Liu and Wang, 2011)) and regression tasks (such as modeling user preference (Ahmed, Low, Aly, Josifovski and Smola, 2011) or estimating stock prices (Dashnyam, Liu, Hsu and Tsai, 2011)). Furthermore, the learning algorithms supported include all the supervised learning algorithms, e.g., Naive Bayes, SVMs, Decision Trees, or k-Nearest Neighbor (see the experimental section, Section 6 for the related experiments), because no restrictive assumptions are made for the details of the learners.

3. Problem Formulation

We term the problem formulation, the "problem of the demanding lord". The idea is that there is a demanding lord that requires a meal every day from his good servant. The servant tries to estimate a ranking or classification of the meals his lord likes, based on his reactions to previous meals. Each day the servant offers a set of meals and gets the full set of reactions from the lord as feedback. The lord, however, changes his preferences randomly. We want to determine how many of the lord's latest answers the servant needs to remember, in order to offer the lord the most satisfactory meals on average over time. It is important to note that we allow the servant to have his own way of learning based on his lord's answers.

The analogy to the actual learning problem is the following:

- The context generating (and causing changes to) the target concept — e.g., the user, or the market stakeholders — $\mathbb{W}$ is the demanding lord.
- The learning system $\mathbb{H}$ is the servant.
- A day identified by its number $d, d \in \mathbb{N}^*$, counted from the beginning of the servant's arrival, is a system iteration.
- The meal G is the information that has to be evaluated by the learned model.
- The preference A of the lord to a meal is the feedback to the system, concerning a given meal. We consider this to be a value taken from a set $\mathbb{A}$. We note that this problem formulation allows application of our method on both regression and classification problems. In classification problems $\mathbb{A}$ is an enumerated set of classes. In a regression problem on $\mathbb{R}$, $\mathbb{A} = \mathbb{R}$. In the case of regression a total ordering operator ($\leq$) should be imposed, i.e., there are some degrees of preference.
- The way the servant learns, or training policy $\mathbb{P}$, is the machine learning algorithm or methodology used by the learning system.

– The number r of the lord's reactions, which the servant remembers when training.
– The period of shift T_s is the time of days (iterations) that (are expected to) pass between two consecutive concept shifts. In the case of random shifts T_s, can be approximated by the expected value of days between two consecutive shifts, given a set of existing observations.

We can differentiate servants from their policy of learning $\mathbb{P}$ and by the number of reactions r they take into account.

The finite-memory servant remembers the last r reactions only.

The all-remembering servant remembers all his lord's reactions. The all-remembering servant is a special case of the finite-memory servant one with $r \to \infty$.

Therefore, a servant can be described as the pair $\mathbb{H} \equiv < \mathbb{P}, r >$. The lord can be described based on the probability distribution $p(d)$ of an occurring shift, over the days elapsed from the last shift: $\mathbb{W} \equiv < p(d) >$. To facilitate the reader, we provide in Table 1 a synopsis of the notation we use in the following paragraphs.

Example 3.1. Following the above analogy and symbols, consider a user $\mathbb{W}$. The user uses an e-mail spam filtering system $\mathbb{H}$, based on the Naive Bayes algorithm $\mathbb{P}$. The spam filter, trying to be somewhat adaptive takes into account for training the reactions of the user $\mathbb{W}$ within the last $r = 30$ days. Thus, $\mathbb{H} =< NaiveBayes, 30 >$.

Each e-mail G is labeled as "spam" or "legitimate (ham)" by the system. Then the user reacts to the label of an e-mail by either agreeing or disagreeing ($A = \{agree, disagree\}$).

50 days after the first use of the system, $\mathbb{W}$ becomes ill and is suddenly only interested in pharmacy related e-mail, which was previously considered "spam". This causes a complete, sudden shift in the user's interests. Another $T_s = 50$ days later the user has gotten well and his interests shift once again.

We make some assumptions that facilitate the representation of the problem:

– The lord $\mathbb{W}$ periodically changes his interests through what we call an *interest shift*, or simply *shift*. This implies that:
$p(d) = 1$, if $d = kT_s, k \in \mathbb{N}^*$ else $p(d) = 0$. Based on Example 3.1, our user's $p(d) = 1$, if $d = 50 \times k, k \in \mathbb{Z}_+^*$, otherwise $p(d) = 0$.
– A shift is radical, so that no information is valid concerning reactions on the previous sets of meals. This makes sure that we know which part of the information we have is useful, based on the knowledge of the last interest shift that has occurred. In our example, when our user becomes ill, he does no longer care about anything he cared before the illness.
– There is no periodicity in the change of interests, i.e., we cannot predict the interests of the lord after a shift, based on previous shifts. Every shift can be considered to be a randomization of interest over meals, independent of the previous randomizations. This makes sure that all information before the last shift is useless, i.e., noise. In our example, once the user gets over his illness, he now gains a completely new set of interests, independent of his original interests (before the illness).

We stress at this point that, even though we make these strict assumptions, we

$\mathbb{W}$	The context/lord.
$\mathbb{H}$	The adaptive system/servant.
r	The memory window of the system/servant.
$\mathbb{T}$	The training set of the system/servant.
$\mathbb{S}$	The valid feedback subset of the training set.
$\mathbb{N}$	The no longer valid feedback subset of the training set.
T_s	The period of the concept/interest shift.
d	The current iteration/day.
ρ	The ratio of the window to the period ($\frac{r}{T_s}$).
$\mathbb{U}_d$	The preference function related to the context/lord for day d.

Table 1. Notation synopsis

show in Section 6 that the method is applicable even in settings where none of these assumptions stand.

For a given day d and a set of offered meals

$$\mathbb{G}_d = \{G_1, G_2, G_3, ..., G_n\}, n > 0$$

the set of the lord's reactions on that day is

$$\mathbb{A}_d = \{A_1, A_2, A_3, ..., A_n\}$$

containing the reactions mapped to each one of the n meals. If $U_d : \mathbb{G}_d \rightarrow \mathbb{A}_d$ is the mapping (model) of meals to reactions for a given day, then we define as *signal* for day d any $< G, A >, G \in \mathbb{G}, A \in \mathbb{A}$ pair where $A = U_d(G)$ and as *noise* any other pair $< G', A' >, G' \in \mathbb{G}, A' \in \mathbb{A}$. In the following paragraphs, for the sake of simplicity, when we refer to feedback A we imply the $< G, A >$ pair, unless otherwise noted. A shift on day d implies that $U_{d-1} \cap U_d = \emptyset$. If no shift occurs, then $U_{d-1} = U_d$ (inertia of the target concept). Based on Example 3.1, the user keeps his interests $U(G)$ until the illness comes up. And then again, right after the illness none of his previous interests stand. Thus, $U_{49}(G) \neq U_{50}(G)$.

In the following elaboration we refer to Figure 2 to visualize the described states. In Figure 1 we provide the explanation-legend of the corresponding symbols. Each given day d_c, the servant $\mathbb{H}$ uses the r last feedback sets (see Figure 2) $\mathbb{A}_{d_c-r}, \mathbb{A}_{d_c-r+1}, ..., \mathbb{A}_{d_c-1}$ to learn, using his training policy $\mathbb{P}$, to estimate meals. We call this set of feedback sets the training set $\mathbb{T}$ of the servant. In a given point it time d_c the servant is trained using only valid information (the white circles in Figure 2), if within the last r days, no shift has occurred. Otherwise, if a shift occurred on day d_s, before the current day d_c, $d_c - d_s \leq r$, then the servant has some no longer valid feedback set $\mathbb{N} \subset \mathbb{T}$ (noise, shown as black circles in Figure 2) and some valid $\mathbb{S} \subset \mathbb{T}$ (signal), and $\mathbb{T} = \mathbb{S} \cup \mathbb{N}$. The first interest shift happens on day $d = T_s$. We start with this assumption of periodicity, to facilitate the formulation of the problem. Later (in Sections 6.1.5, 6.4), we demonstrate that our analysis is also valid for random shift frequency cases.

If $|$'$|$ is the operator of the size of a training (sub)set, then we let $S = |\mathbb{S}|$ and $N = |\mathbb{N}|$ represent the signal magnitude and the noise magnitude of a training set $\mathbb{T}$. We also allow $\mathbb{S} = \emptyset \Rightarrow S = 0, \mathbb{N} = \mathbb{T} \Rightarrow N = r$, when all the training set is not valid any longer because a shift has just occurred. Correspondingly, $\mathbb{N} = \emptyset \Rightarrow N = 0, \mathbb{S} = \mathbb{T} \Rightarrow S = r$, when no change has occurred within the last r days (fore more intuition on why this is the case see Figure 2).

Given the above definitions, we define the signal to noise ratio Z of a given moment in time, as:

$$Z = ln'S - ln'N \tag{1}$$

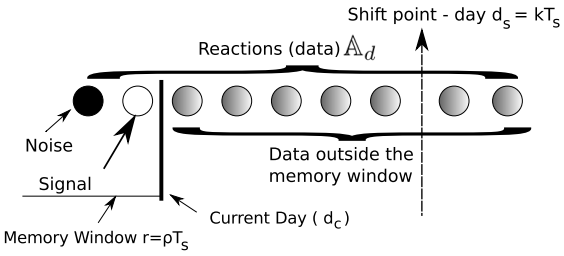


Fig. 1. Explanation of the symbols in the following illustrations (not a valid example).

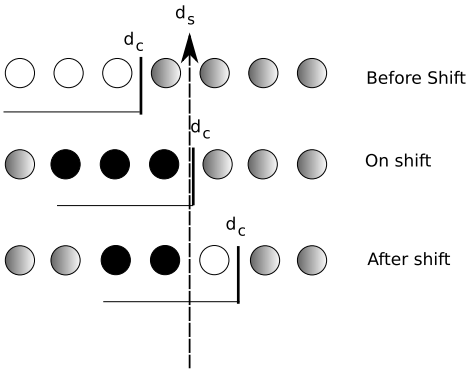


Fig. 2. The validity of training data over time. When the current day is after an interest shift, all data before the interest shift become invalid (i.e., noise).

where $ln'x = \ln(1+x)$ returns for a given x the natural logarithm of $x+1$, to return a value also for $x = 0$.

Let us consider that the servant training set size r is a ratio ρ of the shift period T_s:

$$r = \rho T_s \tag{2}$$

We call this ratio ρ, i.e., the memory window-to-shift period ratio, *characteristic ratio* of a given servant $\mathbb{H}$. In our example, the ratio $\rho = \frac{30}{50} = 0.6$. For a given servant, $\mathbb{H}$ and a given lord $\mathbb{W}$ we support that the servant's average performance on predicting the preferences of the lord is a function $f(\mathbb{W}, \mathbb{H}, \rho)$ or, equivalently, $f(p(d), \mathbb{P}, \rho)$. This means that we consider the performance to be a function of the shift probability distribution, the learning algorithm and the characteristic ratio.

4. Concept Drift and Window Size

We perform an analysis of the effect of the memory (window) size of a learner on its performance, in the presence of concept shift. The analysis is based on the estimation of a function connecting signal-to-noise in the training set to the expected performance of a given learner. The estimation process is described in Section 4.1. Then, given the estimated function, an analysis of the mathematic

relation between the characteristic ratio of a learner and the signal-to-noise ratio, for every iteration of a modeling system with periodic concept shifts, allows the estimation of the average performance of the learning system over time.

The described analysis can be used, in conjunction with concept shift detection methods (e.g., (Gama, Medas, Castillo and Rodrigues, 2004; Patist, 2007)) or various related shift indicators (see (Klinkenberg and Renz, 1998) for an overview of indicators) to optimize the ρ parameter, for a given underlying context $\mathbb{W}$ and learning algorithm.

4.1. The Characteristic Transfer Function of a Learning Algorithm

To describe the $\mathbb{P}$ component of the predictive function $f(p(d), \mathbb{P}, \rho)$, we consider that each learning algorithm is described by a function which indicates the impact of signal-to-noise ratio in the training set to the performance of the algorithm. Given that an algorithm has a minimum performance of $\underline{m}$ and a maximum performance of $\overline{M}$ for a given domain, then we argue that the function that describes the *average* performance $\overline{f}$ as a function of the signal-to-noise ratio Z, is of the form:

$$\overline{f}(Z) = \underline{m} + (\overline{M} - \underline{m}) \frac{1}{1 + b \times exp(-c \times Z)} \tag{3}$$

where $exp(x) = e^x$ and the constants $b \in \mathbb{R}, c \in \mathbb{R}$ are parameters of the *sigmoid* function. We call the $\overline{f}$ function the *characteristic transfer function (CTF)* of the learning algorithm. In the case where $\underline{m} = 0, \overline{M} = 1$, Equation 3 takes the form

$$\overline{f}_N(Z) = \frac{1}{1 + b \times exp(-c \times Z)} \tag{4}$$

which we call the *normal characteristic transfer function (NCTF)* and it represents a normalized version of the CTF.

The intuition behind the sigmoid in Equations 3 and 4 is based on the fact that a (non-trivial) learning algorithm starts to perform well after a certain ratio of good to bad examples has been observed. From that moment on, the performance of the algorithm constantly improves (on average) as the ratio is improved (*monotonicity assumption*), until the point where the best performance is reached. Then, no matter how much the ratio of good to bad examples increases, there is little change, because the algorithm cannot do much better, e.g., due to its generalization property. We consider the CTF to be characteristic of an algorithm for a given dataset. We expect that the sigmoid can be estimated from training sets of varying Z and, then, it can be used as a known function for the given algorithm. We illustrate this property of the function in the experimental section (Section 6).

In existing literature there have been works that estimate aspects of a learning algorithms' performance, e.g., based on the interaction between design (training) and test instance sets (Fukunaga and Hayes, 1989b), or based on the relation between number of training samples and features to performance (Fukunaga and Hayes, 1989a). However, these works do not refer to noisy settings. Other works measure the generalization ability, e.g., of Support Vector Machines (Joachims,

2000) or neural network classifiers (Musavi, Chan, Hummels and Kalantri, 1994). However, these approaches do not deal directly with the presence of noise in the original data and they do not offer a straight-forward way for estimating the performance as a function of the signal-to-noise ratio.

We emphasize that we do not make any specific assumption for the underlying distribution of training instances in the concept space.

The proposed CTF estimation methodology exploits experimental results to estimate the parameters of the CTF discussed in our analytical study (Section 5). If a more accurate formula for the CTF is available, then our estimated function can be simply replaced, without any further consequence to the overall methodology of average performance estimation, described in the following sections, as long as the monotonicity assumption remains in force.

4.2. Average Performance as a Function of Memory Window to Shift Period Ratio

Given the estimation of the CTF, which describes the $\mathbb{P}$ component of the servant, we need to find the relation between ρ and $\overline{f}(Z)$. We examine the following cases:

- The short-memory servant, where $\rho \leq 1$.
- The long-memory servant, where $\rho > 1$.
- The infinite-memory servant, where the performance is the limit of the finite-memory servant performance, when $\rho \to \infty$.

4.2.1. The Short-memory Servant ($\rho \leq 1$)

For the case where $\rho \leq 1$, we can calculate the signal to noise ratio, studying different key iteration intervals as follows:

- In the beginning $d = 0$, $S = 0$, $N = 0$.
- In the interval $0 < d < T_s$, $S = min(d, r)$, $N = 0$. This happens because the maximum number of training instances are r i.e.,

$$S + N = r \Rightarrow N = r - S \tag{5}$$

  Since everything we know so far is suddenly useless, i.e., noise, if $[a]_b$ is the integer part of the division $\frac{a}{b}$ (integer division operator), it stands that:

$$d \in \{d' \,|\, [d']_{T_s} = 0\} \Rightarrow S = 0, N = r \tag{6}$$

- The sum of training instances are r at most; also, everything that is not signal, is necessarily noise. Thus, while $d \in \{d' | d' > T_s, 0 < [d']_{T_s} < T_s\}$ it stands that

$$S = min([d]_{T_s}, r), N = r - S \tag{7}$$

  The function $min()$ is the minimum function.

In Figure 3 we illustrate the case where the characteristic ratio $\rho = 1$, and $T_s = 3$.

Since we are interested on the overall average performance, we will only take into account equations 6, 7. From the above we deduce that S, N are actually $S(d), N(d)$ functions of the current day. Every day d the Z is: $Z(d) =$

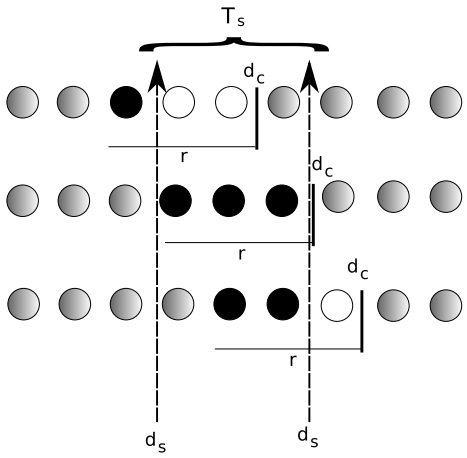


Fig. 3. Three consecutive days of a short-memory servant. $\rho = 1, T_s = 3$.

$ln'(S(d)) - ln'(r - S(d)) = ln'(S(d)) - ln'(\rho T_s - S(d))$. The expected value $\overline{Z}$ of Z is:

$$\overline{Z} = E(Z) = \sum_i Z_i q(Z_i) \tag{8}$$

where $q(Z_i)$ is the probability of occurrence of Z_i and $Z_i, 0 \leq i < r = \rho T_s, i \in \mathbb{N}*$ indicates the possible values of Z.

The possible values of Z, that appear after the first shift, are exactly $r = \rho T_s$ in number, since S is an ρT_s bounded function of $d(modT_s)$ (see Eq. 7). Their values are:

$$Z_i = ln'(i) - ln'(\rho T_s - i), 0 \leq i < \rho T_s$$

The possible values of Z before the first shift, indicated as Z_i^1, are also exactly $r = \rho T_s$ with

$$Z_i^1 = ln'(i) - ln'(0), 0 \leq i < \rho T_s$$

because no noise is present. Over the time of several days (i.e., many iterations), the probability $q(Z_i^1)$ of the Z_i^1 values tends to zero, because there is only one occurrence of the value, regardless of the current day. So for day d, the probability $q(Z_i^1) = \frac{1}{d}$, so when $d \to \infty \Rightarrow q(Z_i^1) \to 0$. In the following analysis, we therefore ignore th Z_i^1 values of Z.

A day d can be modeled related to the last shift that occurred. So, if d is k_1 days after the last shift, which occurred on day $k_0 T_s$, then we can express d as $d = k_0 T_s + k_1$. Then, for a chosen, arbitrarily big $k_0 >> 0$, $k_0 \simeq k_0 + 1$

$$d = k_0 T_s + k_1, k_0 >> 0, 0 \leq k_1 < \rho T_s$$

i.e., after many shifts, all the values $Z_i, i \neq \rho T_s - 1$ have a probability of approximately

$$q(Z_i) = \frac{k_0}{k_0 T_s} \Rightarrow$$

$$q(Z_i) \simeq \frac{1}{T_s} \tag{9}$$

since they appear k_0, or $k_0 + 1$ times (depending on k_1) in $k_0 T_s$ days. For the value $Z_{\rho T_s - 1} \equiv Z_{max} = ln'(\rho T_s) - ln'0$ (which is a *constant*) — i.e., the maximum value of Z — the probability of occurrence of Z_{max} is what is left from the probability mass, if you subtract the probabilities of other values:

$$q(Z_{max}) = 1 - \sum_{i=0}^{\rho T_s - 2} q(Z_i) = 1 - \frac{\rho T_s - 1}{T_s} \Rightarrow$$

$$q(Z_{max}) = 1 - \rho + \frac{1}{T_s} \tag{10}$$

Therefore, Equation 8, gives

$$\overline{Z} = \frac{1}{T_s} \sum_{i=0}^{\rho T_s - 2} Z_i + (1 - \rho + \frac{1}{T_s}) Z_{max} \Rightarrow$$

$$\overline{Z} = \frac{1}{T_s} \sum_{i=0}^{\rho T_s - 2} Z_i - (\rho - \frac{1}{T_s} - 1) Z_{max}$$

Setting

$$R_{tot} = \sum_{i=0}^{\rho T_s - 1} Z_i = \sum_{i=0}^{\rho T_s - 2} Z_i + Z_{max} \tag{11}$$

we get:

$$\overline{Z} = \frac{1}{T_s}(R_{tot} - Z_{max}) - (\rho - \frac{1}{T_s} - 1) Z_{max} \Rightarrow$$

$$\overline{Z} = \frac{R_{tot}}{T_s} - \frac{Z_{max}}{T_s} - \rho Z_{max} + \frac{Z_{max}}{T_s} + Z_{max} \Rightarrow$$

$$\overline{Z} = \frac{R_{tot}}{T_s} + (1 - \rho) Z_{max} \tag{12}$$

Example 4.1. Since from Equation 11, R_{tot} is a function of ρT_s, as is Z_{max}, then the average performance is a function of the ρ and T_s as well. In other words, for a given memory size ratio ρ and a given context generating periodic interest shifts, the expected signal to noise ratio is a constant.

Using the same methodology, we calculate the performance of the system, using Equation 3. Given the fact that performance f is a strictly monotonic function of Z, the unique values of f are exactly the same in number, as the Z possible values. In addition, each value $f(Z_i)$ holds the same probability of appearance as Z_i. This means that, similarly to Equation 8, the expected value of the performance of the *system* is:

$$\overline{f} = E(f(Z)) = \sum_{i=0}^{\rho T_s - 1} f_i q(f_i) \tag{13}$$

where f_i is the i-th possible distinct value of $f(Z)$, with $q(f_i) \equiv Z_i$ and, given Equation 3:

$$f_i = \underline{m} + (\overline{M} - \underline{m}) \frac{1}{1 + b \times exp(-c \times Z_i)}$$

Thus, for $\overline{f}$, we have the following equation.

$$\overline{f} = \sum_{i=0}^{\rho T_s - 1} \left(\left(\underline{m} + (\overline{M} - \underline{m}) \frac{1}{1 + b \times exp(-c \times Z_i)} \right) q(f_i) \right)$$

Performing associatively multiplication, and taking into account the fact that $\sum_{i=0}^{\rho T_s - 1} q(f_i) = 1 \Rightarrow \underline{m} \sum_{i=0}^{\rho T_s - 1} q(f_i) = \underline{m}$, we have:

$$\overline{f} = \underline{m} + (\overline{M} - \underline{m}) \sum_{i=0}^{\rho T_s - 1} \left(\frac{1}{1 + b \times exp(-c \times Z_i)} q(f_i) \right) =$$

which, using Equations 9 and 10, gives:

$$\overline{f} = \underline{m} + (\overline{M} - \underline{m}) (\frac{1}{T_s} \sum_{i=0}^{\rho T_s - 2} \frac{1}{1 + b \times exp(-c \times Z_i)} +$$

$$+ (1 - \rho + \frac{1}{T_s}) \frac{1}{1 + b \times exp(-c \times Z_{max})})$$

which with Equation 4, gives:

$$\overline{f} = \underline{m} + (\overline{M} - \underline{m}) \left(\frac{1}{T_s} \sum_{i=0}^{\rho T_s - 2} \overline{f}_N(Z_i) + (1 - \rho + \frac{1}{T_s}) \overline{f}_N(Z_{max}) \right) \tag{14}$$

Example 4.2. Maximization of the average performance $\overline{f}$ of a learning system with a specific training policy (learning algorithm), given a context with periodic interest shifts, can be achieved by merely changing the parameter ρ ($0 < \rho \leq 1$).

4.2.2. The Long-memory ($\rho > 1$) and the Infinite-memory ($\rho \to \infty$) Servant

In the case of the long-memory servant (see Figure 4), the main difference is the fact that the upper bound of the noise N changes for a given $\mathbb{W}$, since from Equations 6 and 7 the only thing that changes is $r = \rho T_s$. After the first shift, there is no occasion where $N = 0$ (see the upper part of Figure 4).

The probability mass for Z_{max}, $q(Z_{max}) = \frac{1}{T_s}$, since now Z_{max} appears only once from a shift to the next. The new Z_{max} has a value of:

$$Z_{max} = ln'(T_s - 1) - ln'((\rho - 1)T_s + 1) \tag{15}$$

which is upper bounded by $ln'T_s$, no matter how large ρ is. Thus, the equation of average performance, when $\rho > 1$ becomes:

$$\overline{f} = \underline{m} + (\overline{M} - \underline{m}) \left(\frac{1}{\rho T_s} \sum_{i=0}^{\rho T_s - 1} \overline{f}_N(Z_i) \right) \tag{16}$$

The expected signal-to-noise $\overline{Z}$ is lowered, due to the omnipresent noise N. The same stands for performance. As $\rho \to \infty$, Equation 15 indicates that its value will be dominated by the effect of noise. This means that the average performance provided by Equation 16 will indefinitely decrease to asymptotically reach the minimum value, as $\rho \to \infty$ (Infinite-memory servant).

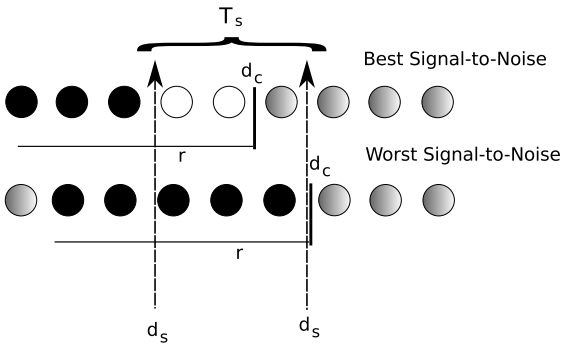


Fig. 4. The best and worst signal-to-noise ratios of a long-memory servant. $\rho = \frac{5}{3}, T_s = 3$.

To summarize, the expected performance of the system when the characteristic ratio ρ exceeds 1, will deteriorate. A previous case study, performed in (Giannakopoulos and Palpanas, 2010a), on the relation between regression performance and memory window ratio, appears to validate this assumption. In the case where the characteristic ratio tends to infinity, the average performance of the system over time will tend to be minimum, due to the unbounded effect of noise, over a bound signal remains contribution to the average performance.

4.2.3. Optimization of the Performance

We argue that the optimization of a system, consists of the following process steps:

- Estimation of the characteristic transfer function for a given learning algorithm and domain. To attain this aim one needs a set of training instances with different preference values (i.e., classes or ordered preference values) $A_i \in \mathbb{A}$ covering as much as possible $\mathbb{A}$. In this step, the focus is on detecting the relation between signal, noise and learning algorithm performance.
- Detection of the interest shift period. In our experimentation section (Section 6), we also use the estimated average interest shift period to see whether the estimation process stands for random interest shifts.
- Optimization of the ρ parameter. We do not elaborate on the optimization process within this work.
- Building of a classification model, using the calculated window size based on the optimized ρ.

For the optimization process there are two important considerations. The first is that the values of the estimated CTF and the true function of performance should be as linearly correlated as possible. If the correlation is strong, then the optimization process will give near-optimal results. The second is that, if there is a requirement for a maximally *exact* estimation of the average performance, we need not only a collinear CTF but a CTF with minimum error, as error can be defined by e.g., the absolute difference between the estimated and actual value of performance for all possible Z values.

If the above process stands, then all problems that can be expressed through the problem of the demanding lord can have an a-priori estimation of perfor-

mance for any given algorithm. The additional effort is to estimate the characteristic transfer function of the learning algorithm and calculate the expected shift period, which of course may be non-trivial tasks.

We stress that, even on online scenaria, the CTF calculation and the *expected* shift period can be obtained by periodic estimation, based on a sample of data (see also the German Market, Electricity and Chess dataset experiments in Sections 6.4 and especially Section 6.6). Thus, the learning of the CTF in a real-setting can be conducted by simply selecting a training subset of the data and synthetically inserting noise at various levels and various (arbitrary) shift points. This means that knowledge of the real shift period in the data is not required beforehand. The experimental results show that the cases with a very small number of training instances (i.e., S tends to 0 and N tends to 0) do not adversely affect our approach.

5. Experimental Setup and Considerations

To validate the force of the assumptions we have made, we perform a set of experiments on different tasks. The first task is a regression-based task, with a target function representing user interest which is shifted from time to time, either in a periodic (Section 6.1.4) or a random (Section 6.1.5) fashion. The second task is a boolean concept attainment task, based on the approach of the STAGGER system (Schlimmer and Granger, 1986). The third task (Section 6.3) is classification task, based on a moving plane, as described in the SEA system (Street and Kim, 2001). The next task is applied on a real-world dataset, concerning the financial environment of Germany over time (Section 6.4), where some quantifying factors of business environment are mapped to the current state of the market (which is the drifting concept). Finally, we examine the case where the data is not available a priori and our method is based on an initial (small) training sample from the data stream. For this task we use two real datasets (Elec2 and Chess datasets).

The purpose of the experiments is twofold:

- First, we want to check whether estimation of the characteristic transfer function (CTF) of an algorithm can be achieved. To do this, for each dataset we perform a 5-fold (or 10-fold, as indicated) cross-validation for the estimation process (using as data the outcome of 10-fold classification experiments on the dataset). We stress that this 5-fold (or 10-fold) process is meant to estimate the CTF curve and not learn a concept or classify instances. The estimation training set is used to estimate the CTF, which is in turn validated on the corresponding test set. To validate the model we use a linear correlation (Pearson) test, where a high correlation value with statistical significance will indicate a good estimator.
- Second, we want to determine whether the analytic estimator of signal-to-noise converges to the actual average signal-to-noise of the system in the random shifts case. If this is true, then the analytic estimator can be used for online optimization of ρ, recalculated after every detected shift.

For each task, we describe the dataset used and the evaluation methodology for the learning algorithm, we perform the modeling of the CTF for two algorithms and we validate the CTF estimator concerning the mean performance

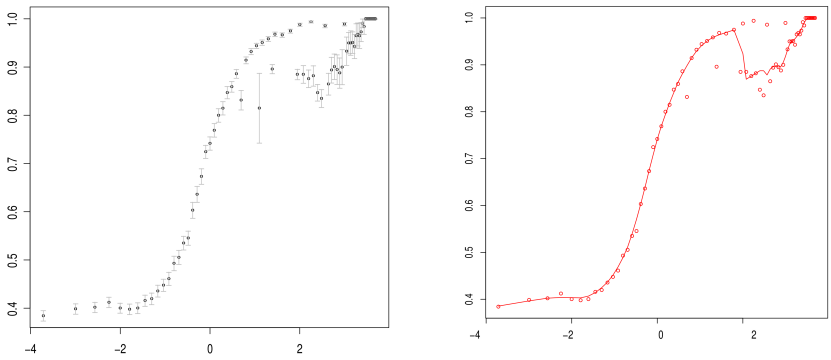


Fig. 5. Mean Performance per signal-to-noise ratio (LogSN): Means plot with standard deviation bars (left) and LOWESS regression plot (right).

(accuracy unless otherwise noted) estimation. Then, we provide the correlation between the estimation and the real average performance of the system.

Before we proceed with the evaluation, we give an example of a Signal-to-Noise to Mean Performance graph — drawn from a single test case — in Figure 5. The left figure indicates the average performance per signal to noise, including also confidence interval bars within a 95% confidence level. Due to the fact that a signal-to-noise value may appear more or less times, there are estimations of average performance that are quite uncertain. Thus, to view the overall tendency of average performance we also use LOWESS regression (Cleveland, 1981) (with a span parameter of 0.03) to indicate a better estimate of average performance per signal-to-noise value (right part of Figure 5). The figure illustrates that a sigmoid underlying function possibly relates Z to performance $\overline{f}$. In our experiments, we will try to search for a good-enough sigmoid that describes the average performance based on signal-to-noise. For the LOWESS regression calculation we also use Z values that have enough support, i.e., have appeared enough times to have a standard error below 0.05 for the estimation of their mean performance.

Searching for a Good Sigmoid

Given an equation of sigmoid type (see Equation 3), we need to identify the parameters that best describe a set of observed data points. In our case, the data points are measured performance values for given signal-to-noise ratios.

The search in the parameter space is performed by a genetic algorithm (Goldberg, 1989), searching for an approximate good set of parameters[2]. The genetic algorithms can maximize a function value by changing a set of parameters related to that value. They are certain to converge to an optimal solution given time, but they can also provide sub-optimal good results after a limited time of iterations. They are most useful when the relation between the parameters and

[2] We also applied non-linear regression alternatives for the search, but the methods failed to converge in some cases and, thus, we were forced to try genetic algorithms.

the function is non-linear and difficult to study analytically. The definition of the problem in the genetic algorithm context requires the definition of:

– The genome: a set of parameters that will be changed and combined based on genetic operators to form candidate solutions. The genome is actually a set of alleles, the elements of which are mapped one-to-one to the parameters of the model. Thus, in the sigmoid CTF equation

$$\overline{f}(Z) = \underline{m} + (\overline{M} - \underline{m}) \frac{1}{1 + b \times exp(-c \times Z)}$$

we are searching for good values of $\underline{m}, \overline{M}, b, c$, for which we have used corresponding alleles.

– A fitness function: a function that uses the genome to evaluate how good a candidate solution is. The Genetic Algorithm aims to maximize the fitness function. In our case the fitness function is based on the Kolmogorov-Smirnov (KS) goodness-of-fit D statistic (Massey Jr, 1951). We note that, even though there are other versions of the KS test with more power (e.g., see (Harter, Khamis and Lamb, 1984)), in this case we simply use the D statistic as an indication of fitness, not really using the test significance values.
Consider that we have a set of data points (x, y), where each x indicates a signal-to-noise ratio and each y the corresponding measured average performance for this x. Then, we estimate a second set of points $(x, \overline{f}(x))$. We would like to know whether the distribution of values of y is actually different within statistical error from the one of $\overline{f}(x)$.
The Kolmogorov-Smirnov test statistic D is expected to have a low value if two sets of samples from distributions are more likely to originate from the same underlying distribution. In fact, it measures the maximum vertical distance between the Cumulative Distribution Function of a hypothesized underlying distribution of a set of samples to an ideal distribution. In our experiments, we try to optimize the parameters of the sigmoid CTF equation, so that the D value between the y values and $\overline{f}(x)$ distributions is minimized. This aims to find a sigmoid that can be used for the best possible approximation of the average performance and not only for a function correlated to the true CTF, as we discussed also in Section 4.2.3.

As a result of the above and of the nature of the D statistic, our search for a good sigmoid is reduced to the search of good enough parameters that minimize the maximum distance between the point-wise values of the Cumulative Distribution Function of the measured average system performance and the estimated average system performance for a given set of signal-to-noise values.

To determine whether the sigmoid estimation is indeed a good estimator even given unseen values of signal-to-noise we perform a five-fold cross-validation of our sigmoid estimation process. In every fold we use $\frac{4}{5}$ of the data points as the training set and the rest as the test set. The training set is used to determine the CTF and the test set to determine the collinearity (through a Pearson test) between the estimation and the real values of the performance with the given CTF. A high collinearity value indicates that the CTF is a good estimator and can be used for optimization, where collinearity between the estimated and true values are of concern. The fact, however, that we search for a maximally similar function, through the use of the Kolmogorov-Smirnov test, allows even a good approximation of the actual average performance value.

Name	*Type*	*Brief description of data and task*	*Source*
ENS	Synthetic	User modeling (regression) task, using complex data (string and multi-dimensional)	(Giannakopoulos and Palpanas, 2010a)
STAGGER	Synthetic	Boolean concept attainment (classification)	(Schlimmer and Granger, 1986)
SEA	Synthetic	Moving plane (classification) task.	(Street and Kim, 2001)
German Market	Real	Market state identification (classification) task.	(Ifo Institute, 2010)
Elec2	Real	Electrical power price (classification) task.	(Harries and Wales, 1999)
Chess	Real	Chess game outcome (classification) task.	(Zliobaite, 2010)
Airline Delay	Real	Airline delay estimation (regression) task.	(Ikonomovska, Gama and Deroski, 2011)

Table 2. Description of the used datasets and corresponding tasks.

To be able to have robust results when estimating the CTF, as well as when testing the average performance, we wanted to take into account only values of Z that have enough support in the data points, which means that the standard error of the mean estimation is within a small interval (see also Figure 5 for the problem of mean estimation). Thus, before generating the training and test sets we make sure that we keep averages that have a standard error of mean below 0.05. As a reminder, the standard error of the mean σ_μ is measured by

$$\sigma_\mu = \frac{\sigma}{\sqrt{N}}$$

where σ is the standard deviation of the samples and N the number of those samples. We note that the estimation process execution time is in the order of a few seconds per fold in all the following experiments.

6. Experimental Evaluation

In the following paragraphs we illustrate the results of our experiments on three synthetic and four real datasets, briefly described in Table 2. We cover the cases of both periodic and random interest shifts, and we also examine the effectiveness of our method on data where not all the theoretic assumptions (see Section 3) hold. We also cover cases where none of the assumptions hold and there is a limited training subset before the full data become iteratively available (Section 6.6).

We stress that this section aims to validate whether our theoretic findings are applicable in a variety of settings. The use of this method in a real system would imply the selection of a specific shift detector, which may seriously affect

the performance of the system. Thus, we try to remain impartial by considering optimal shift detection in the following experiments.

6.1. Experiments on the ENS Dataset

For our first dataset, we use the case of an adaptive entity-subscription system (Giannakopoulos and Palpanas, 2009), which ranks, for each subscriber, information chunks based on their importance, as determined by previous feedback from the same subscriber. Thus, the approach is a feedback-based one. In the following paragraphs we describe the setting and corresponding task in more depth.

The dataset is based on the setting of a subscription service of an Entity Name System (ENS) (Bouquet, Stoermer and Bazzanella, 2008; Palpanas, Chaudhry, Andritsos and Velegrakis, 2008). The service allows people to get informed about changes on data items they have subscribed to. The data items are sets of free-form string key-value pairs, where each set offers information about a real-world entity, such as a person, a location, etc. These changes are caused by editors of an open community.

Each subscriber ranks the information received using a qualitative scale based on importance: unnecessary, useful and critical. The qualitative scale maps to a quantified interest level: -1.0, which indicates unnecessary information, 1.0, which indicates useful information and 2.0, which indicates critical information. The ranking acts as feedback to the system. The system takes into account both the type of the change on an entity (deletion, update, etc.) but also the content (e.g. which is the new value of an added attribute), representing all this change information into vector space (see (Giannakopoulos and Palpanas, 2009) for more). The system models the preferences of the subscriber as a regression model on this vector space, based on ϵ-SVR (Vapnik, 1998).

The dataset is synthetic and has two aspects. The first aspect (Section 6.1.1) is a set of changes happening to the entities, based on such editor classes, as malevolent, benevolent and administrator. The second aspect (Section 6.1.2), which is the one used for the subscriber modeling, is the feedback from users to the generated changes. From this feedback a learning algorithm tries to estimate the preference function of the subscriber. In the following paragraphs we elaborate on those aspects. We chose to use this dataset due to its complex but understandable modeling of preference and performance, that allows for generic application.

6.1.1. ENS User Behavior Data

As behavior of the editors that change the entities' data, we generate a number of changes' descriptions D — 10000 instances split into sets of 1000 instances to provide for 10-fold validation. To generate this kind of dataset, we randomly create changes based on a selection from the following editors behaviors: benevolent user changes, system administration changes and malevolent user changes. The probabilities of emission per editor profile and change type are elaborated on in Table 3.

User type (Prob.)	*Change type*	*Probability*
Benevolent (0.95)	Attribute change (normal)	0.60
	Attribute insertion	0.30
	Attribute deletion	0.10
Sys.admin.(0.03)	Entity merge	0.45
	Entity split	0.45
	Entity deletion	0.10
Malevolent (0.02)	Attribute change (abnormal)	0.70
	Attribute deletion	0.30

Table 3. ENS user behavior: probability distribution of change emission

6.1.2. Entity Subscribers' Feedback Data

The second part of this dataset consists of subscriber feedback. We consider a few representative cases of subscribers to minimize the evaluation overhead, while providing useful insight on the adaptivity of the system:

– a subscriber who is interested in the deletion of attributes, to make sure no information concerning the entities he has subscribed to are missing. This scenario is expected to mostly use the type of the change description as a discriminating factor of importance.
– a subscriber mostly interested in the changes of names entities. This subscriber scenario is expected to mostly use the content of a change description as a discriminating factor of importance.
– a subscriber who is interested in whether any entity he has subscribed to has its "isDeceased" status changed. This scenario aims to represent the difficulty of users interested in specific attribute *values*.
– a subscriber who has the role of a validator of data in OKKAM, who wants to be aware of abnormal changes in default attributes, such as deletion or change with an abnormal value. This subscriber scenario is expected to use both the type and the content of a change description as a discriminating factor of importance.

A more detailed description of what each user finds interesting and critical can be found in Table 4. All changes not noted within a profile are considered uninteresting for the profile. The profiles have been chosen so that they are require different kinds of information to be determined, concerning either the type or the content of the change.

Before the evaluation of the learning, we use the learning methodology and produce corresponding results, also supplying feedback for every step. This process is reiterated for every subscriber scenario for the whole set of change data.

6.1.3. Evaluation Methodology for the ENS dataset

To evaluate the learning performance we "emit" changes to the supposed user — in groups of ten — and measure how well the system adapts to the feedback. We consider that the user feeds back the system after every new emission, by indicating the importance of all the items in the last group. We define the Rate

Subscriber	*Importance*	*Description*
Type-based	Critical	Attribute deletion.
	Interesting	Entity deletion.
Attribute name-based	Critical	Any change concerning an attribute that contains the string "name".
	Interesting	(None)
Attribute name-value pair-based	Critical	Attribute change or insertion on "isDeceased" attribute, with a new value of "true".
	Interesting	Attribute change or insertion on "isDeceased" attribute, with a new value of "false".
Complex	Critical	Default attribute (some attributes in the ENS are considered default — e.g., the name of a person entity — while all the others non-default) update or insertion with an abnormal value.
	Interesting	Default attribute deletion or normal update.

Table 4. Profile Descriptions. Note: All changes not noted within a profile are considered uninteresting for the profile.

of Acceptable Errors measure, elaborated below, as an appropriate measure of performance for the task.

The system performance for a given change emission is set to be the number of times a ranking error has exceeded 0.5. Given our interest values, errors beyond this 0.5 margin *may* cause an error. Errors below this margin cannot cause an error by themselves. So the performance is *the percentage of the importance estimation in a given set that have their absolute error below* 0.5. Thus, a value of 1.0 in performance indicates a ranking that is ideal, while a value of 0.0 indicates a ranking that will have several errors. We call this measure *Ratio of Acceptable Errors (RAE)* and we use it as an equivalent of the classification accuracy for a regression problem: acceptable errors are considered good estimates. The formula, for a given set $\mathbb{D}$ of change descriptions, their corresponding sequence of importance estimations $\tilde{\mathbb{I}}$ and actual importance values $\mathbb{I}$, is:

$$RAE(\tilde{\mathbb{I}}, \mathbb{I}_0) = 1.0 - \frac{\sum_{i \in (1,\ldots,|\mathbb{I}|)} \lfloor min(|\tilde{\mathbb{I}}(i) - \mathbb{I}(i)|, 0.5) + 0.5 \rfloor}{|\mathbb{I}|} \tag{17}$$

where $\tilde{\mathbb{I}}(i)$, $\mathbb{I}(i)$ is the i-th element of the corresponding sequence, $\lfloor x \rfloor$ is the floor operator, $|\mathbb{X}|$, gives the number of elements of a sequence $\mathbb{X}$, $|x|$ is the absolute value of a number x and $min(x, y)$ is the minimum function.

6.1.4. Estimation of CTF for Regression — Periodic Change

In Table 5 we see the estimation correlation to the actual performance function. We provide the quantiles of the distribution of results from a 5-fold cross-validation, as described above. We also calculate the mean correlation over all folds, which is found in bold in the table. In the table we illustrate the results for various ρ values and we also perform a run with different T_s to test whether the method results are overall well, regardless of parameters.

The results indicate clearly that there is a consistently very strong linear

SVR							
Setting		**Correlation Quantiles**					
T_s	ρ	Min	1st Q.	Median	**Mean**	3rd Q.	Max
100	0.50	0.98	0.98	0.98	**0.98**	0.99	0.99
100	0.85	0.96	0.97	0.97	**0.97**	0.98	0.98
100	1.00	0.97	0.97	0.98	**0.98**	0.98	0.99
100	1.15	0.98	0.98	0.99	**0.98**	0.99	0.99
200	1.00	0.86	0.87	0.88	**0.89**	0.89	0.94
Linear Regression							
Setting		**Correlation Quantiles**					
T_s	ρ	Min	1st Q.	Median	**Mean**	3rd Q.	Max
100	0.50	0.80	0.89	0.96	**0.92**	0.97	0.98
100	0.85	0.79	0.84	0.86	**0.87**	0.91	0.97
100	1.00	0.85	0.85	0.87	**0.88**	0.88	0.96
100	1.15	0.83	0.84	0.89	**0.90**	0.95	0.98

Table 5. Periodic Shift: Pearson Correlation between performance through CTF estimation and actual performance values.Note: All correlations are with a p-value of $<< 0.01$.

correlation between the estimation of the performance for every Signal-to-Noise and the actual data performance. All the results are within the 99% confidence level (p-value which is $<< 0.01$). Thus, we conclude that the sigmoid estimation process can determine an approximation function that predicts successfully the relation between Z and $\overline{f}$.

It is important to note that the estimation process is effective, *irrespective of different learning methodologies* (namely Support Vector Regression using a radial basis kernel vs. simple linear regression).

6.1.5. Estimation of CTF for Regression — Random Change

To further examine whether our estimation process stands, we checked the case of random period of interest shift. In this case the shift can take place at any point between $T_s \pm (0.5 \times T_s)$ (uniform sampling). The process returns more possible Z values. We note, as we can see in Table 6, that the consistently strong correlation still stands, regardless of the underlying learning method.

We found, through experiments that the CTF may be better expressed by the (equivalent) form:

$$\overline{f}(Z) = \underline{m} + (\overline{M} - \underline{m}) \frac{1}{1 + b \times exp(-c \times (Z - d))} \tag{18}$$

This form adds a parameter d indicative of the horizontal position shift. This addition merely helps the estimation algorithm to determine the parameters of the sigmoid.

In the following experiments we have added the d parameter. The use of the d parameter allowed over 0.4 improvement in the Pearson correlation value between the estimation of the CTF and the actual data points.

SVR							
Setting		**Correlation Quantiles**					
T_s	ρ	Min	1st Q.	Median	**Mean**	3rd Q.	Max
100	0.50	0.98	0.98	0.99	**0.99**	0.99	0.99
100	0.85	0.96	0.98	0.98	**0.98**	0.98	0.99
100	1.00	0.96	0.97	0.98	**0.98**	0.98	0.99
100	1.15	0.96	0.98	0.98	**0.98**	0.98	0.99
200	1.00	0.92	0.96	0.97	**0.96**	0.98	0.98
Linear Regression							
Setting		**Correlation Quantiles**					
T_s	ρ	Min	1st Q.	Median	**Mean**	3rd Q.	Max
100	0.50	0.78	0.85	0.95	**0.90**	0.96	0.97
100	0.85	0.72	0.86	0.92	**0.87**	0.92	0.94
100	1.00	0.86	0.87	0.93	**0.92**	0.97	0.99
100	1.15	0.82	0.83	0.94	**0.90**	0.95	0.96

Table 6. Random Shift: Pearson Correlation between performance through CTF estimation and actual performance values. Note: All correlations are with a p-value of $<< 0.01$.

6.2. Experiments on the Boolean Concept Dataset

The second dataset we use is based on the STAGGER method evaluation dataset (Schlimmer and Granger, 1986), which is traditionally used in the concept drift domain. The experiment is performed as follows. The problem is a problem of classifying 100 randomly generated instances per iteration as either belonging to the current iteration's target concept or not. The instances are objects described based on three dimensions: size, which can be small, medium or large; color, which is either red, green, or blue; shape, which is either square, circular, or rectangular. In the beginning of the task, the set of 120 training instances is generated randomly from the possible variations of objects. Each instance is labeled according to the following rules:

- In iterations 1–40 only *small, red* objects are considered to belong to the target concept.
- In iterations 41–80 objects that are either *green or circular* are considered to belong to the target concept.
- In iterations 81–120 objects that are either *medium or large* are considered to belong to the target concept.

In every iteration i the training algorithm has access to the first i training instances. On that same iteration, the testing is performed on 100 randomly generated instances, which are labeled according to the current target concept. That is, for the *test data*, on every iteration the labeling is coherent with the current target concept.

The results on this dataset, shown in Table 7, indicate the consistently high correlation of the estimation to the true performance.

Naive Bayes							
Setting		**Correlation Quantiles**					
T_s	ρ	Min	1st Q.	Median	**Mean**	3rd Q.	Max
40	0.50	0.89	0.94	0.95	**0.94**	0.95	0.96
40	0.85	0.87	0.88	0.89	**0.90**	0.89	0.95
40	1.00	0.97	0.97	0.98	**0.98**	0.98	0.98
40	1.15	0.88	0.90	0.90	**0.91**	0.90	0.96
J48 Decision Tree							
Setting		**Correlation Quantiles**					
T_s	ρ	Min	1st Q.	Median	**Mean**	3rd Q.	Max
40	0.50	0.83	0.86	0.96	**0.92**	0.96	0.97
40	0.85	0.84	0.87	0.88	**0.89**	0.91	0.95
40	1.00	0.92	0.95	0.97	**0.96**	0.98	0.98
40	1.15	0.88	0.91	0.91	**0.92**	0.93	0.96

Table 7. STAGGER Dataset: Correlation in 10-fold cross-validation between performance through CTF estimation and actual performance values. Note: All correlations are with a p-value below 0.05.

6.3. Experiments on the Moving Plane dataset: The SEA Concept Drift Dataset

The third synthetic dataset we use comes from the Streaming Ensemble Algorithm (SEA) (Street and Kim, 2001) (variations of this dataset have been used in several papers, e.g., (Wang, Fan, Yu and Han, 2003)). This dataset is created as follows. 50000 Random training points are created in a 3-dimensional space, using three features $< f_1, f_2, f_3 >, f_i \in \mathbb{R}$. The points are divided into four blocks. In each block, a data point belongs to (is labeled as) "class 1", if $f_1 + f_2 \leq \theta$, where θ is assigned values $8, 7, 9, 9.5$ correspondingly to the blocks. 10% class noise is further inserted in each block of data. Then, 10000 instances are generated as test instances. These instances are not labeled yet.

To evaluate the performance of a system over time, we use a 500-instance step on each iteration. The system uses a window of recent (previous) instances and tries to learn the concept of "class 1". The evaluation is performed on the test instances on every iteration. These instances are correctly assigned labels based on which the current block is (and the corresponding θ value) per iteration.

The results of 5-fold cross validation on this dataset, shown in Table 8, indicate the consistently high correlation of the estimation to the actual performance. The results we show were extracted from a Support Vector Machine Classifier and a Naive Bayes Classifier. Similar results were extracted using a Decision Tree and a Nearest Neighbor classifer.

6.4. Experiments on Real Datasets

In this section we examine whether the CTF can be adequately well estimated, based on real world datasets. We use two ways to estimate the CTF. First, we use the whole dataset (using the German Market dataset). Then, we also try to

Naive Bayes							
Setting		**Correlation Quantiles**					
T_s	ρ	Min	1st Q.	Median	**Mean**	3rd Q.	Max
12500	0.40	0.83	0.90	0.92	**0.91**	0.93	0.96
12500	0.80	0.71	0.86	0.87	**0.85**	0.90	0.92
12500	1.00	0.87	0.88	0.89	**0.90**	0.91	0.96
12500	1.04	0.69	0.73	0.87	**0.82**	0.88	0.91
SVM							
Setting		**Correlation Quantiles**					
T_s	ρ	Min	1st Q.	Median	**Mean**	3rd Q.	Max
12500	0.40	0.78	0.84	0.84	**0.85**	0.90	0.90
12500	0.80	0.70	0.74	0.80	**0.80**	0.85	0.90
12500	1.00	0.76	0.81	0.84	**0.84**	0.86	0.93
12500	1.04	0.66	0.83	0.89	**0.85**	0.92	0.94

Table 8. SEA Dataset: Correlation in 10-fold cross-validation between performance through CTF estimation and actual performance values. Note: All correlations are with a p-value below 0.01.

estimate the CTF using few instances from a long set of instances (the Electricity dataset).

6.4.1. The German Market Dataset

The first real world dataset we use is based on the Ifo Business Survey, which describes some aspects of the business climate in Germany (Ifo Institute, 2010). The data are based on questionnaires filled in by more than 7000 participating members, indicating the view of the companies on business climate, current business situation and business outlook. The dataset contains one record per month from January, 1991 to February, 2010 (230 records). Each record contains six measurements representing the *balance* values and the *index* values of the business situation, the business outlook and the business climate. The balance value of the current business situation is the difference in percentage of the responses "good" and "poor" in the questionnaires; the balance value of the business outlook (expectations) is the difference in percentage shares of the responses "more favorable" and "more unfavorable". The business climate is a transformed (actually, geometric) mean of the balances of the business situation and the business expectations. The data also includes corresponding index values, normalizing the balances to the average of the year 2000 (which is mapped to the index value 100). Based on the signs of the balances the study classifies the state of the market as "boom", "downswing", "recession", "upswing" (also see Table 9). We note that this dataset functions as a "stress test" of our methodology: the dataset contains very few records. From the theoretical point of view, such analysis provides some insight on the minimum amount of data necessary for the method to function. From the application point of view, there exist settings where the sampling rate can be low (e.g., when social studies are required to get the data, as is the case for the German Market Data). Furthermore, there is a question of whether our method performs well when really few instances are present (e.g., during the

	Situation Negative	*Situation Positive*
Expectations Positive	Upswing	Boom
Expectations Negative	Recession	Downswing

Table 9. The state of the market as related to business situation and business expectations balance signs.

SVM							
Setting		**Correlation Quantiles**					
$\overline{T_s}$	$\rho(r)$	Min	1st Q.	Median	**Mean**	3rd Q.	Max
8.46	0.59 (5)	0.82	0.87	0.95	**0.91**	0.95	0.96
8.64	1.04 (9)	0.91	0.91	0.93	**0.93**	0.96	0.96
8.48	3.75 (30)	0.90	0.91	0.95	**0.94**	0.96	0.97
NN							
Setting		**Correlation Quantiles**					
$\overline{T_s}$	$\rho(r)$	Min	1st Q.	Median	**Mean**	3rd Q.	Max
8.46	0.59 (5)	0.84	0.87	0.91	**0.90**	0.93	0.93
8.64	1.04 (9)	0.89	0.91	0.91	**0.92**	0.93	0.98
8.48	3.75 (30)	0.96	0.97	0.97	**0.97**	0.98	0.99

Table 10. German Market Dataset: Correlation (5-fold validation) between performance through CTF estimation and actual performance values.

early phases of data collection). This dataset allows us to examine whether and how the presented approach can be applied in such extreme settings.

We consider that a system tries to learn the characteristics of the current state of the market. Thus, if the market in iteration d is in state "Upswing", the system considers that the last few iterations (within its training window) model this state. The system then tries to predict whether the market in the next iteration is the same state, based on its features. This implies that every change in the dataset from one class to another (e.g., "Upswing" to "Boom") in consecutive iterations is a shift point.

This dataset caused a problem for the evaluation of the CTF estimation, due to the fact that 5-fold cross-validation was not possible. This was caused by the few iterations and the few samples for different values of Z, which caused some folds to lack testing set, if the weakly supported Z values were ignored.

To avoid this pitfall we have applied evaluation using the 5x2cv method(Dietterich, 1998) guidelines. The idea is that we perform 5 times 2-fold cross validation and each time the data of each fold are picked randomly from the original dataset. Based on this method, each time we learn the CTF based on only 115 samples and we evaluate the CTF on the remaining 115 samples. For the evaluation, we measure the correlation of the estimated performance for the Z values in the test set to the actual performance in the test set. The results of the evaluation are illustrated in Table 10 and indicate very high correlation between the estimated and the true performance value for a given Z.

6.4.2. The Chess Dataset

The Chess dataset (Zliobaite, 2010) consists of 533 instances, obtained from chess.com portal. According to the document describing the dataset "the data consists of game records of one player over a period from 2007 December to 2010 March. A player has a rating, which changes depending on his/her results achieved (the higher is the rating, the stronger is the player). A player is developing skills over time, besides engages into different types of tournaments and competitions. The rating and the type of game determine how the system selects an opponent. This is where the concept drift is expected. The task is to predict if the player will win or lose based on the setting. There is natural problem of delayed labelling, the winner is known only after the game is finished. In turn based chess one game might last even for several months".

The dataset contains the following attributes: the Outcome (lost, won, or draw) which is used as the instance label; Moves (the number of moves until the end of the game); White/black (the color of the pieces for the player); Current rating; Oponent's rating; Type (type of the game: personal, tournament, championship); Speed (days allowed per move); Start date. The data were sorted by start date, but otherwise the date attribute itself was ignored for the classification problem. About 37% of the instances were losses, 57% were wins and the remaining 6% were draws.

For this dataset we selected the first 50 samples (less than 10% of the whole set) to estimate the CTF. The average shift frequency in the dataset is extremely low: every 2.20 iterations we have a shift of the current state (label). The estimated average period within the first 50 instances is 2.11, which is used in the CTF estimation. This essentially means that there are too few points to fully regenerate a sigmoid representative of the corresponding algorithm for a whole range of Z values. On the other hand, we do not need to construct such a detailed CTF, since the levels of noise in this specific dataset are within the observed limits.

The results, when estimating the CTF for linear SVM — based on the LibLINEAR library (Fan, Chang, Hsieh, Wang and Lin, 2008) — and Voted Perceptron (Freund and Schapire, 1999), are still good (average correlation well over 0.75). We omit these results for brevity, since they do not offer new findings.

6.4.3. The Electricity Dataset

The Electricity market dataset was first described (as Elec2) by Harries (Harries and Wales, 1999). It contains 45312 instances, drawn from May 7, 1996 to December 8, 1998, with each instance mapped to a half-hour interval. The class label indicate whether the price of electricity was increased or decreased, as related to a moving average of the last 24 hours. As in previous works (Baena-García, del Campo-Ávila, Fidalgo, Bifet, Gavaldà and Morales-Bueno, 2006), we have considered this problem as trying to predict the label of the class attribute for the next 30 minutes.

The estimation of the CTF was performed, using only the first 1000 instances (about 2% of the whole set of instances). The average period of this sample was 7.11 iterations, while the true (over the whole dataset) was 6.82.

In Table 11 we illustrate that the Pearson correlation between the estimated and true performance of the NN and JRip algorithms on the Electricity dataset is high.

NN							
Setting		**Correlation Quantiles**					
$\overline{T_s}$	$\rho(r)$	Min	1st Q.	Median	**Mean**	3rd Q.	Max
7.11	0.71 (5)	0.75	0.77	0.78	**0.80**	0.82	0.88
JRip							
Setting		**Correlation Quantiles**					
$\overline{T_s}$	$\rho(r)$	Min	1st Q.	Median	**Mean**	3rd Q.	Max
7.11	0.71 (5)	0.61	0.77	0.90	**0.83**	0.91	0.93

Table 11. Electricity Dataset: Correlation (10-fold validation) between performance through CTF estimation and actual performance values for first 1000 instances.

6.4.4. The Airline Delay Dataset

The Airline Delay Dataset contains flight arrival and departure details for all the commercial flights within the USA, from October 1987 to April 2008, provided in the Data Expo competition of 2009. The dataset contains around 120 million records. We used a version (from Elena Ikonomovska) whcih has been cleaned and records were sorted according to the arrival/departure date (year, month, and day) and time of flight. From this dataset we used the records of flights from 1988 that involved the Washington Dulles Internation airport (airport code: IAD). There records were 100.161, containing information related to the departure and the arrival of a flight (time and location), total distance, and the airline.

In order to define when concept drift occurs in the dataset we performed clustering over the whole set of instances. We used k-means clustering, with 5 clusters (k=5). The idea is that the clustering can detect underlying concepts in the instance space. Thus, we can use the information of the cluster succession in our iterative learning to determine whether a concept shift has occurred (similarly to the STAGGER dataset). The average period of shift detected was 3.13 iterations. Therefore we used the values $\rho \in \{3, 6, 12\}$ to perform CTF estimation using the 5x2foldCV method (to avoid problems with under-represented Z values).The CTF estimation aimed at the rule-based M5P regression method(Wang and Witten, 1996) and a "decision tree"-based method(Kohavi, 1995), as they have been implemented in the WEKA platform(Hall, Frank, Holmes, Pfahringer, Reutemann and Witten, 2009, v. 2.7.4).

In this setting, we chose to use the Mean Absolute Error (MAE) as an indicator of performance. Thus, we provide an example of application when the performance measure is unbounded and, in this case, an error estimate. The higher the Mean Absolute Error of the regression estimation, the worse the estimation is. In order to provide a performance measure that would increase when the performance is better, and ideally within the $[0, 1]$ interval, we represented the system performance as a function of MAE:

$$f(MAE) = 1.0 - exp(-MAE), MAE \geq 1 \tag{19}$$

In our experiments each iteration we used the 100 last instances as training and estimated the next delay value. The results, shown in Table 12 indicate that the

Decision Tree							
Setting		**Correlation Quantiles**					
$\overline{T_s}$	$\rho(r)$	Min	1st Q.	Median	**Mean**	3rd Q.	Max
3.13	0.96 (3)	0.67	0.68	0.79	**0.78**	0.83	0.92
3.13	1.92 (6)	0.70	0.77	0.77	**0.78**	0.81	0.87
3.13	3.83 (12)	0.74	0.81	0.85	**0.83**	0.87	0.89
Rule-based M5P							
Setting		**Correlation Quantiles**					
$\overline{T_s}$	$\rho(r)$	Min	1st Q.	Median	**Mean**	3rd Q.	Max
3.13	0.96 (3)	0.50	0.51	0.54	**0.62**	0.71	0.84
3.13	1.92 (6)	0.65	0.75	0.85	**0.81**	0.87	0.93
3.13	3.83 (12)	0.74	0.78	0.81	**0.80**	0.81	0.84

Table 12. Airline Delay Dataset: Correlation quantiles between performance through CTF estimation and actual performance values.

CTF is a good approximation of the relation of Z to performance. We notice that for values for low ρ the correlation is not extremely (especially in the M5P case). This is caused by the fact taht the period T_s is very low: for low ρ the range of possible Z values is minimal (3-4 values). Thus, there exist few samples to measure correlation between true and estimated performance values. For higher ρ values, this problem is mitigated.

6.5. Convergence between estimated and real average performance

In this section we study whether the estimated and real average performance of a *demanding lord system* (DLS), converge over time. We perform an estimation of the sigmoid as indicated before and we keep the best performing estimation function (in terms of collinearity to the actual performance). Using this CTF we follow a system over time recalculating on every iteration:

- the average period of the shift. The calculation is essentially performed when a shift is indicated. In our scenaria we know when the shift occurs, however in a real world application a detection method would need to be applied.
- the estimated performance, based on the sigmoid and the formula of estimation of the average performance.
- the actual performance, based on the feedback.

Given the above information we plot graphs indicating the relation between iteration and absolute difference between the estimation and the actual value (absolute error). We call this absolute difference the *delta* of performance. To determine whether this delta is reduced over time, indicating convergence, we measure the Spearman and the Pearson correlations, indicating rank and linear correlation correspondingly between iteration and delta. If their values are negative (ideally strongly negative), this would indicate that *the absolute error is*

Synthetic dataset: ENS Dataset - 6000 iterations			
Setting		Spearman	Pearson
Regression	100-50	-0.663204	-0.4264592
Regression	100-100	-0.9421365	-0.5005253
Regression	100-150	0.6391099	-0.1948143
Synthetic dataset: Boolean Concept Dataset - 6000 Iterations			
Setting		Spearman	Pearson
Bayes	40-20	-0.3022561	0.-0.272744
Bayes	40-40	-0.9611965	-0.482552
Bayes	40-60	0.8951005	0.3195209
Real dataset: Business Climate - 212 Iterations			
Setting		Spearman	Pearson
SVM	Random-5	-0.1777558	-0.3815536
SVM	Random-9	-0.3168430	-0.4193718
SVM	Random-15	-0.1914687	-0.3843884

Table 13. Correlation of iteration number and delta (10-fold Validation)

reduced over time. The system takes into account on every iteration the average period, based on the shifts so far, in the case of the real dataset, where there is no fixed period for the shift.

We perform 10-fold validation (10 experiment runs), also augmenting the number of iterations during which we examine the system, for the synthetic datasets. For example, for the Boolean Concept (STAGGER) dataset, we expanded the iterations to 6000. For the convergence checking we used only two out of three synthetic datasets (STAGGER and ENS), covering the cases of classification and regression[3]. The results are illustrated in Table 13. We also provide figures for the different ρ values ($\rho < 1$, $\rho = 1$,$\rho > 1$) for each dataset: ENS Dataset, Figure 6; Boolean Concept dataset, Figure 7 ; Business Climate dataset, Figure 8. The vertical axis of the figures corresponds to the delta, while the horizontal axis to the iteration number. The dashed line indicates the linear tendency (linear regression line) of the delta over iterations.

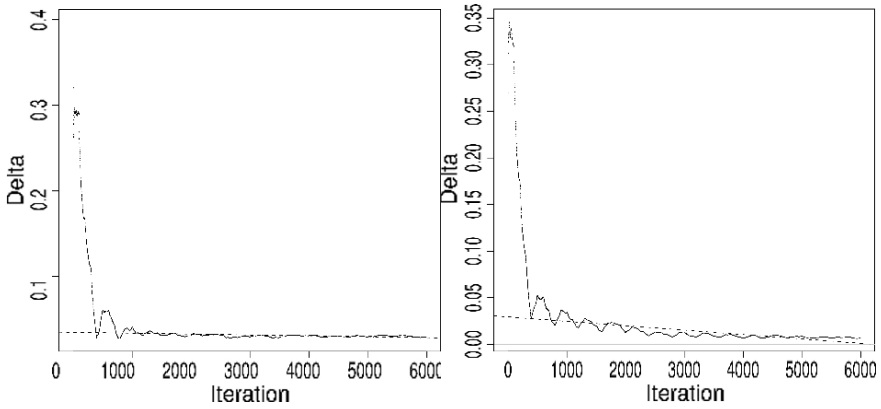
We see that, even though when $\rho \leq 1$ the performance estimation converges, this does not stand when ρ gets higher values. This comes to validate the fact that the estimation cannot be optimal when either of the two following problems appear.

- If there is periodicity in the drift (see also Section 3) and $\rho > 1$ then we may mistake signal for noise.
- The CTF estimated in collinear to but not a precise approximation of the real CTF.

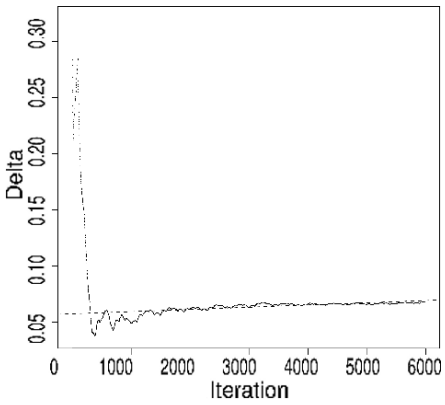
In the cases where the estimation does not converge both problems appear. This is also clear for the Airline Delay Dataset, in Figure 9.

On the other hand, it is very interesting that the system does well even in the case of the random shift (e.g., Business climate dataset, in Figure 8), even

[3] The results of the SEA dataset are similar to those of the STAGGER dataset.



(a) $\rho < 1$ (b) $\rho = 1$



(c) $\rho > 1$

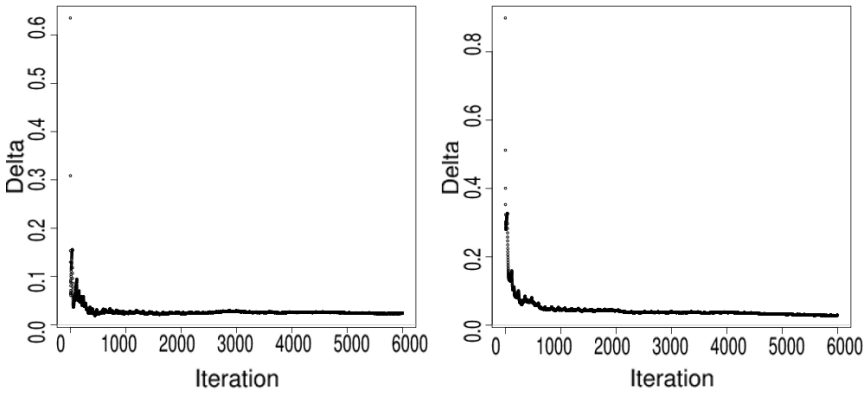
Fig. 6. ENS Dataset: Convergence (10-fold validation) of the average delta. Dashed line: linear regression line fitting the observations.

though it shows that complete randomness in the shift period causes jumps in the value of delta at several points in time.

6.6. Other real datasets — Limited Training

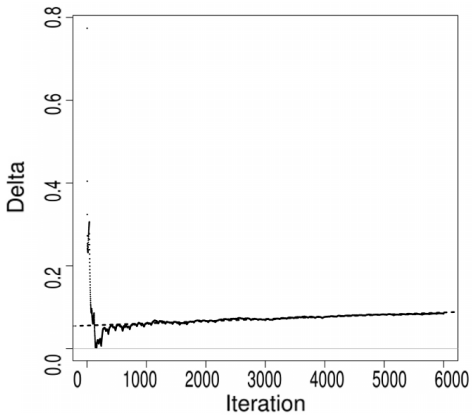
In this section we provide the delta curves for the Electricity Dataset (Figure 10 for NN, Figure 11 for Naive Bayes) and the curves for the Chess Dataset (Figure 12 for the Linear SVM, Figure 13 for the Voted Perceptron). We describe these results separately from the previous dataset due to a methodological change: the datasets were processed in a streaming fashion, i.e., in one pass. Thus, we have used the CTF and average period times as they were estimated from a few, early instances (see Sections 6.4.2, 6.4.3). This corresponds to the real-case scenario, where not all data are available a priori.

In Figures 10, 11, 12, 13, we examine the different cases of ρ. We note that for these experiments the theoretic prerequisites we discuss in our study do not



(a) $\rho < 1$

(b) $\rho = 1$

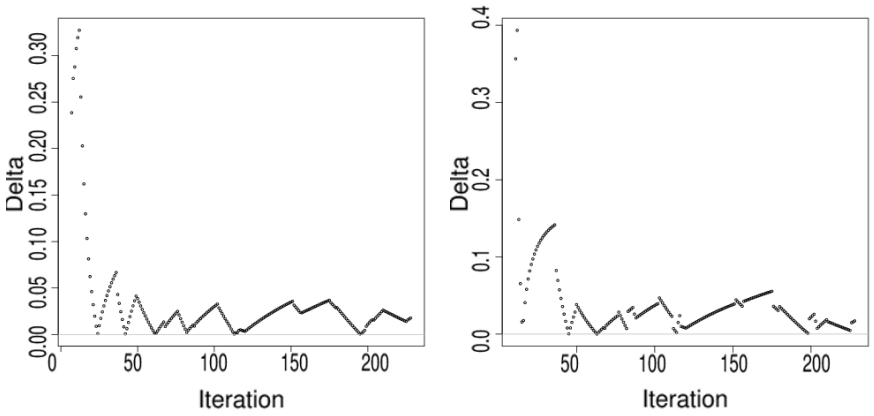


(c) $\rho > 1$

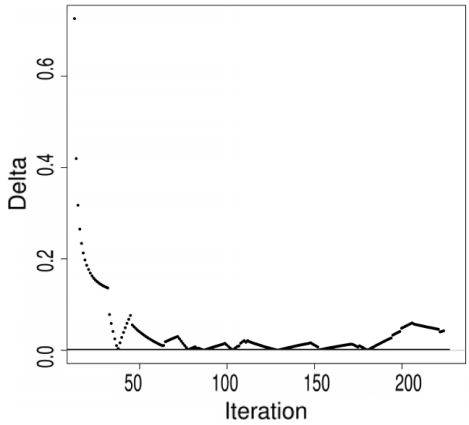
Fig. 7. Boolean Concept Dataset: Convergence (10-fold validation) of the average delta. Dashed line: linear regression line fitting the observations.

hold. However, the estimation process does converges over time (though, not always to zero). The absolute error is on average around 0.05, and in some cases as low as 0.01 (cf. all cases of Figure 12). This indicates that the estimation of average performance can still be precise, even if the theoretic prerequisites do not hold. It would be interesting to theoretically study the impact of removing each prerequisite to the error bounds of the model estimations. We also see, in Figure 13, that the speed of the delta convergence does not appear to be collinear to ρ — i.e., increasing or decreasing ρ does not guarantee a better estimate. Finally, we note that the first few estimation values may bias the correlation of error and time strongly, as we can see in Figures 13a, 13c.

In order to test whether the estimation of the shift period T_s is critical to the performance of the system, we tested whether providing wrong Ts hinders the estimation of the CTF. The CTF is estimated without apparent problems



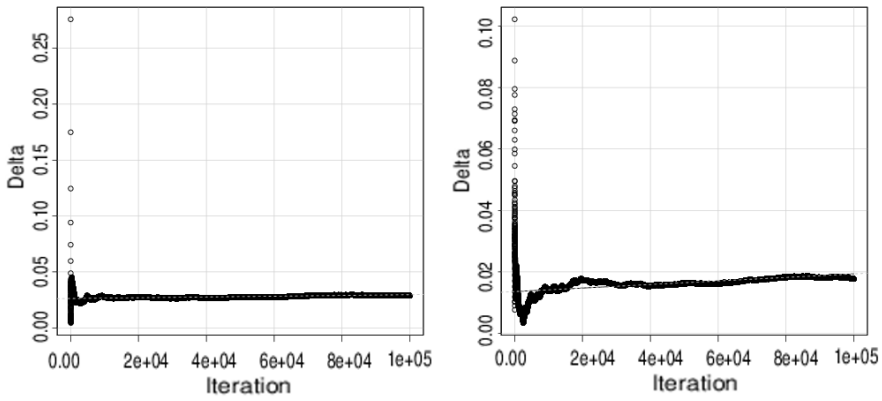
(a) $\rho < 1$ (b) $\rho = 1$



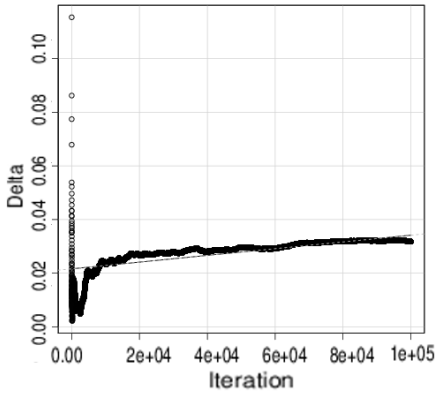
(c) $\rho > 1$

Fig. 8. Business Climate Dataset: Convergence (10-fold validation) of the average delta. Dashed line: linear regression line fitting the observations.

(correlation is high). However, we map wrong Z values (which are calculcated based on the period) to performance values. This essentially means that the CTF is not a good estimate: it either over- or under-estimates the relation between Z and performance. Since the CTF does not map the true values to Z values, the estimation error is increased. In a real, difficult scenario (we used the Electricity dataset case, with the Nearest Neighbor algorithm) where the minimum and maximum performance values range from 0.40 to 0.60, changing the period up to almost 2 orders of magnitude (from 7.11 to 240) caused an error increase from 0.05 to 0.07 (40% increase) to the average performance estimation. Given the fact that the possible performance variation is 0.20, this is not a negligible loss of estimation accuracy. In a case where the range of the CTF is broad (i.e., the gap between the classifier optimal and worst performance in a dataset is big) the



(a) $\rho < 1$ (b) $\rho = 1$



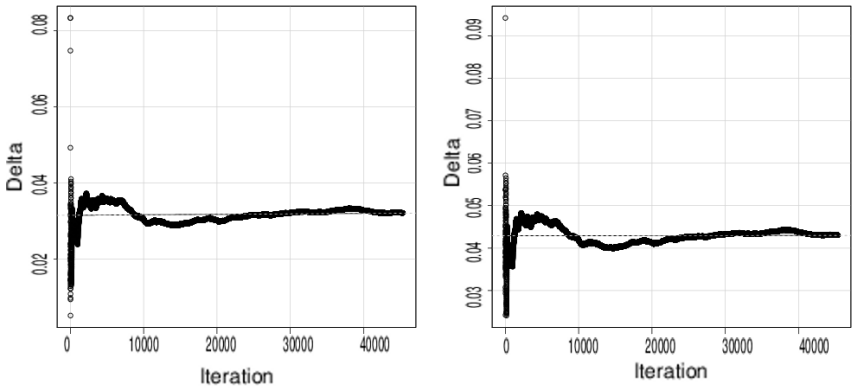
(c) $\rho > 1$

Fig. 9. Airline Delay Dataset: Convergence of the average delta for the Decision Tree algorithm. Dashed line: linear regression line fitting the observations.

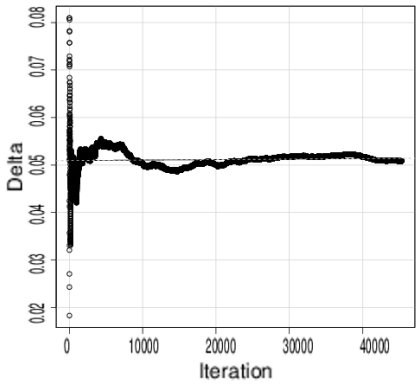
expected degradation of the results would be much higher. Thus, the impact of the period estimation error is connected to the CTF value range and increases as the period estimation error increases.

7. Discussion and Conclusions

Summarizing what we have seen in the course of this study, we have shown that one can estimate a characteristic transfer function, connecting signal-to-noise in the training set of a learning algorithm to the average performance of the algorithm. Given this CTF, one can use a closed-form estimation function for the average performance of a learning system after infinite time, in the presence



(a) $\rho < 1$ (b) $\rho \approx 1$

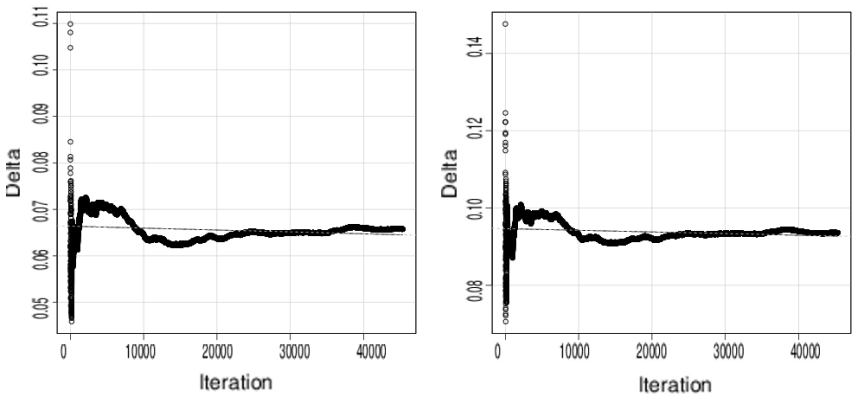


(c) $\rho > 1$

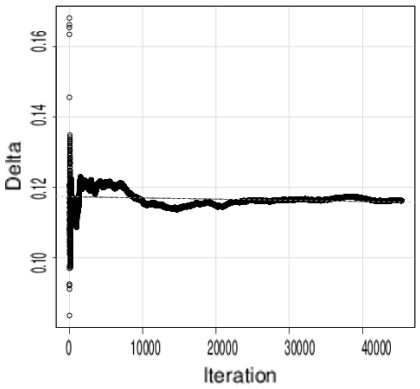
Fig. 10. Electricity Dataset: Convergence of the delta for Nearest Neighbor. Black curve: Sample-based period estimation; Dashed line: linear regression line fitting the observations.

of concept shifts. Even in cases where not all the assumptions of our closed-form estimator stand, e.g., when we know just the average value for the period of change, the system still provides a good estimate of performance as a function of the "memory window"-to-"shift period" ratio. This means that, on a partial memory online learning system, we can estimate a good memory window for a given series of instances. We can even closely predict the average performance, if the CTF estimation is good and our memory window is not bigger than the average concept shift period.

What we also noted in our experiments is that the D value of the Kolmogorov-Smirnov test we used in our search for a good sigmoid CTF and the corresponding confidence interval are highly related to the accuracy of the estimator. This leads us to consider, in the future, how one can define bounds in the estimation error,



(a) $\rho < 1$ (b) $\rho \approx 1$

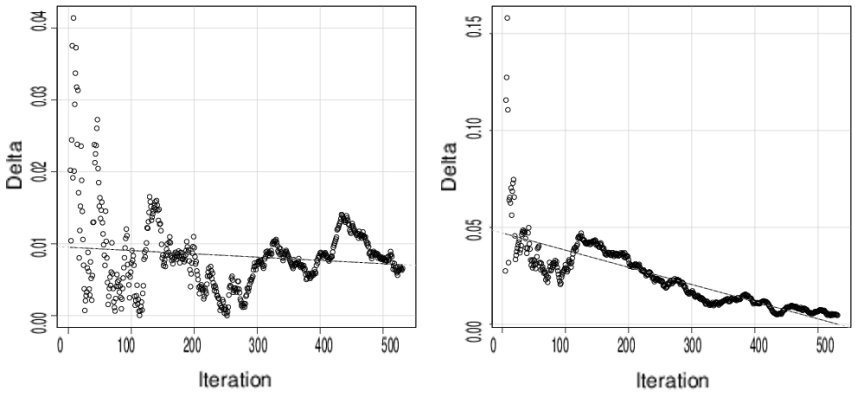


(c) $\rho > 1$

Fig. 11. Electricity Dataset: Convergence of the delta for Naive Bayes. Black curve: Sample-based period estimation; Dashed line: linear regression line fitting the observations.

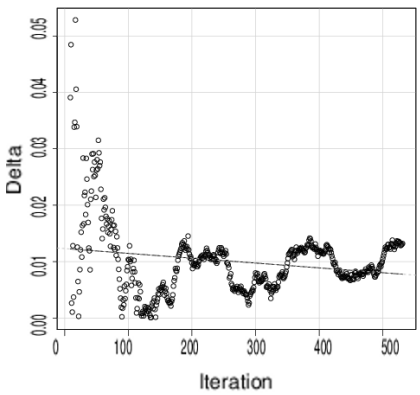
given D, or such measures as the chi-square statistic between the estimated and actual CTF sample values.

The analysis we have performed offers a basis that allows further investigation of the learning algorithms from a signal-to-noise-response perspective. For a given dataset, calculating a good CTF allows for the estimation of performance, without exhaustive experiments. Furthermore, in cases where we know the distribution of signal-to-noise we expect to find in a dataset, we can estimate the average performance of a system by finding a weighted average of the CTF values. We also plan to apply the optimization steps, described in Section 4.2.3 and measure the performance of such a system against state-of-the-art methods of learning in the presence of concept drift. Our method would be interesting

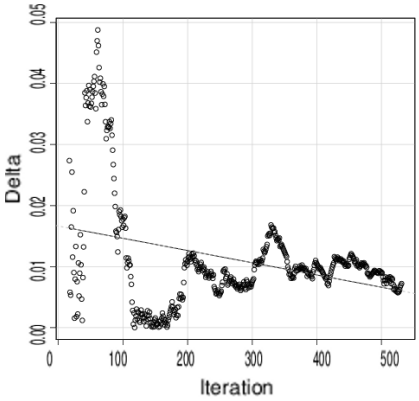


(a) $\rho = 0.95$

(b) $\rho = 1.90$



(c) $\rho = 3.80$

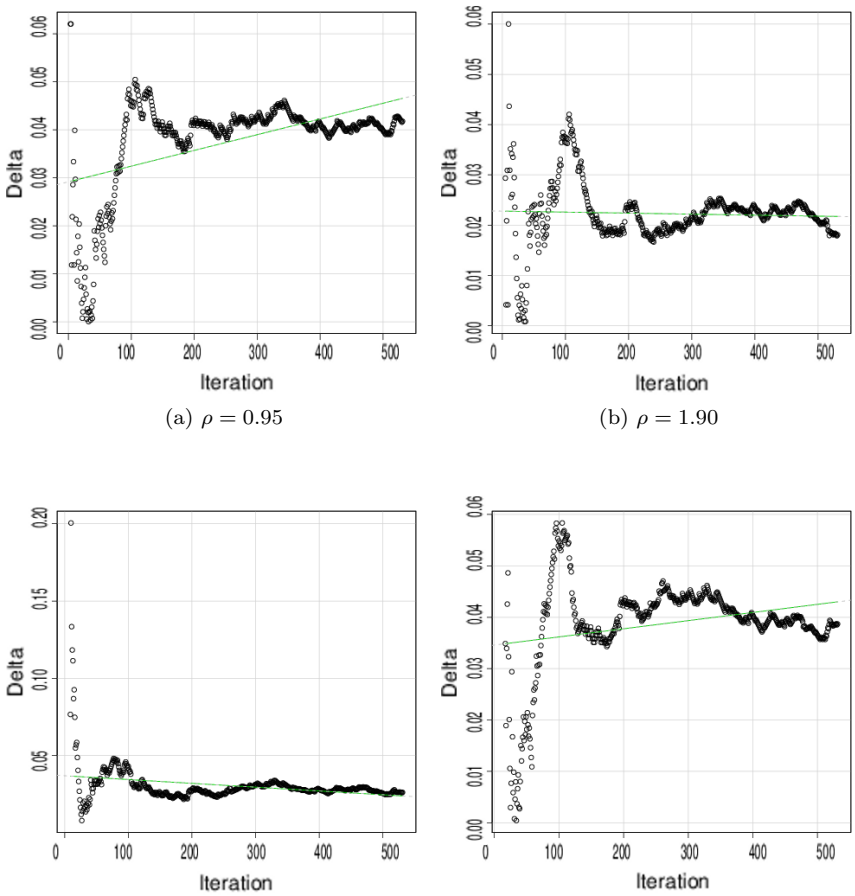


(d) $\rho = 7.60$

Fig. 12. Chess Dataset: Convergence of the delta for LibLINEAR. Black curve: Delta over time (iterations); Dashed line: linear regression line fitting the observations.

to apply on ensemble methods, possibly optimizing individual learners' memory windows.

Taking a different point of view, there is the question of what aspects of an algorithm the CTF parameters describe. For example, the d parameter may indicate stability in the presence of noise. Other questions that arise are the following. Is there always a collapse point and what does this imply for an algorithm? Can one determine dataset-independent parameters that will allow the classification of learning algorithms based on their behavior in the presence of noise? And does the presented methodology stand also for the case of sequential learning? These appear to be promising next steps of the research described in this study and constitute part of our future work.



(a) $\rho = 0.95$ (b) $\rho = 1.90$

(c) $\rho = 3.80$ (d) $\rho = 7.60$

Fig. 13. Chess Dataset: Convergence of the delta for Voted Perceptron. Black curve: Delta over time (iterations); Dashed line: linear regression line fitting the observations.

Acknowledgements. The authors would also like to thank Prof. Alberto Valli of the University of Trento for useful advice and insight on the estimation of the sigmoid function. This work was partially supported by the FP7 EU Large-scale Integrating Project OKKAM "Enabling a Web of Entities" (contract no. ICT-215032). See http://www.okkam.org.

References

Ahmed, A., Low, Y., Aly, M., Josifovski, V. and Smola, A. (2011). Scalable distributed inference of dynamic user interests for behavioral targeting, *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, pp. 114–122.

Angluin, D. (1988). Queries and concept learning, *Machine learning* **2**(4): 319–342.

Baena-García, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavaldà, R. and Morales-Bueno, R. (2006). Early drift detection method, *ECML PKDD 2006 Workshop on Knowledge Discovery from Data Streams*.

Bouquet, P., Stoermer, H. and Bazzanella, B. (2008). An entity name system (ENS) for the semantic web, *European Semantic Web Conference*, pp. 258–272.

Cleveland, W. S. (1981). LOWESS: a program for smoothing scatterplots by robust locally weighted regression, *American Statistician* p. 5454.

Crabtree, B. I. and Soltysiak, S. (1998). Identifying and tracking changing interests, *International Journal on Digital Libraries* **2**(1): 38–53.

Dashnyam, B., Liu, Y.-C., Hsu, P.-Y. and Tsai, Y.-T. (2011). Real time prediction of closing price and duration of b2b reverse auctions, *Knowledge and Information Systems* .
URL: *http://www.springerlink.com/content/p302748t360w4047/*

Dietterich, T. (1998). Approximate statistical tests for comparing supervised classification learning algorithms, *Neural computation* **10**(7): 1895–1923.

Domingos, P. and Hulten, G. (2000). Mining high-speed data streams, *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, pp. 71–80.

Fan, R., Chang, K., Hsieh, C., Wang, X. and Lin, C. (2008). Liblinear: A library for large linear classification, *The Journal of Machine Learning Research* **9**: 1871–1874.

Fdez-Riverola, F., Iglesias, E., Díaz, F., Méndez, J. and Corchado, J. (2007). Applying lazy learning algorithms to tackle concept drift in spam filtering, *Expert Systems with Applications* **33**(1): 36–48.

Fidalgo-Merino, R. and Nunez, M. (2011). Self-adaptive induction of regression trees, *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **33**(8): 1659–1672.

Freund, Y. and Schapire, R. (1999). Large margin classification using the perceptron algorithm, *Machine learning* **37**(3): 277–296.

Fukunaga, K. and Hayes, R. (1989a). Effects of sample size in classifier design, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **11**(8): 873–885.

Fukunaga, K. and Hayes, R. (1989b). Estimation of classifier performance, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **11**(10): 1087–1101.

Gama, J., Medas, P., Castillo, G. and Rodrigues, P. (2004). Learning with drift detection, *Lecture Notes in Computer Science* pp. 286–295.

Giannakopoulos, G. and Palpanas, T. (2009). Adaptivity in entity subscription services, *Proceedings of ADAPTIVE2009*, Athens, Greece.

Giannakopoulos, G. and Palpanas, T. (2010a). Content and type as orthogonal modeling features: a study on user interest awareness in entity subscription services, *International Journal of Advances on Networks and Services* **3**(2).
URL: *http://www.iit.demokritos.gr/ ggianna/Publications/adaptiveSubscription.pdf*

Giannakopoulos, G. and Palpanas, T. (2010b). The effect of history on modeling systems' performance: The problem of the demanding lord, *ICDM 2010*, IEEE.
URL: *http://www.iit.demokritos.gr/ ggianna/Publications/ICDM2010.pdf*

Goldberg, D. (1989). *Genetic Algorithms in Search and Optimization*, Addison-wesley.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten, I. (2009). The weka data mining software: an update, *ACM SIGKDD Explorations Newsletter* **11**(1): 10–18.

Harries, M. and Wales, N. (1999). Splice-2 comparative evaluation: Electricity pricing.

Harter, H., Khamis, H. and Lamb, R. (1984). Modified Kolmogorov-Smirnov tests of goodness of fit, *Communications in Statistics-Simulation and Computation* **13**(3): 293–323.

Helmbold, D. P. and Long, P. M. (1994). Tracking drifting concepts by minimizing disagreements, *Machine Learning* **14**(1): 27–45.
URL: *http://dx.doi.org/10.1007/BF00993161*

Hulten, G., Spencer, L. and Domingos, P. (2001). Mining time-changing data streams, *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM New York, NY, USA, pp. 97–106.

Ifo Institute (2010). Ifo business survey - the ifo business climate for germany since january 1991. http://www.cesifo-group.de/portal/page/portal/ifoHome/a-winfo/d6zeitreihen/15reihen/_reihenkt.

Ikonomovska, E., Gama, J. and Deroski, S. (2011). Learning model trees from evolving data streams, *Data Mining and Knowledge Discovery* **23**(1): 128168.

Ikonomovska, E., Gama, J., Sebastião, R. and Gjorgjevik, D. (2009). Regression trees from data streams with drift detection, *Discovery Science*, Springer, pp. 121–135.

Joachims, T. (2000). Estimating the generalization performance of an SVM efficiently, *International Conference on Machine Learning*, Citeseer, pp. 431–438.

Kiefer, J. (1953). Sequential minimax search for a maximum, *Proceedings of the American Mathematical Society* **4**(3): 502–506.

Klinkenberg, R. and Renz, I. (1998). Adaptive information filtering: Learning in the presence of concept drifts, *Learning for Text Categorization* pp. 33–40.

Kohavi, R. (1995). The power of decision tables, *Machine Learning: ECML-95* pp. 174–189.

Koychev, I. and Lothian, R. (2005). Tracking drifting concepts by time window optimisation, *Proceedings of AI-2005, the Twenty-fifth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, Citeseer, pp. 46–59.

Koychev, I. and Schwab, I. (2000). Adaptation to drifting user's interests, *Proc. of ECML2000 Workshop: Machine Learning in New Information Age*, Citeseer, pp. 39–45.

Kuh, A., Petsche, T. and Rivest, R. (1990). Learning time-varying concepts, *Proceedings of the 1990 conference on Advances in neural information processing systems 3*, Morgan Kaufmann Publishers Inc., p. 189.

Kuncheva, L. and Žliobaitė, I. (2009). On the window size for classification in changing environments, *Intelligent Data Analysis* **13**(6): 861–872.

Lam, W., Mukhopadhyay, S., Mostafa, J. and Palakal, M. (1996). Detection of shifts in user interests for personalized information filtering, *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, p. 325.

Lazarescu, M., Venkatesh, S. and Bui, H. (2004). Using multiple windows to track concept drift, *Intelligent Data Analysis* **8**(1): 29–59.

Liu, W. and Wang, T. (2011). Online active multi-field learning for efficient email spam filtering, *Knowledge and Information Systems* .
URL: *http://www.springerlink.com/content/173321g2qg2u0q61/*

Maloof, M. and Michalski, R. (1999). Selecting examples for partial memory learning, *Machine Learning* **41**(1): 27–52.

Maloof, M. and Michalski, R. (2004). Incremental learning with partial instance memory, *Artificial intelligence* **154**(1-2): 95–126.

Massey Jr, F. (1951). The Kolmogorov-Smirnov test for goodness of fit, *Journal of the American Statistical Association* **46**(253): 68–78.

Mitchell, T., Caruana, R., Freitag, D., McDermott, J. and Zabowski, D. (1994). Experience with a learning personal assistant, *Communications of the ACM* **37**(7): 80–91.

Musavi, M., Chan, K., Hummels, D. and Kalantri, K. (1994). On the generalization ability of neural network classifiers, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **16**(6): 659–663.

Núñez, M., Fidalgo, R. and Morales, R. (2007). Learning in environments with unknown dynamics: Towards more robust concept learners, *The Journal of Machine Learning Research* **8**: 2595–2628.

Palpanas, T., Chaudhry, J. A., Andritsos, P. and Velegrakis, Y. (2008). Entity data management in OKKAM, *DEXA Workshops*, pp. 729–733.

Patist, J. (2007). Optimal window change detection, *Data Mining Workshops, 2007. ICDM Workshops 2007.*, IEEE, pp. 557–562.

Ramon, J., Driessens, K. and Croonenborghs, T. (2007). Transfer learning in reinforcement learning problems through partial policy recycling, *Machine Learning: ECML 2007* pp. 699–707.

Reinke, R. and Michalski, R. (1988). Incremental learning of concept descriptions: A method and experimental results, *Machine intelligence* **11**: 263–288.

Schlimmer, J. and Granger, R. (1986). Incremental learning from noisy data, *Machine learning* **1**(3): 317–354.

Scholz, M. and Klinkenberg, R. (2007). Boosting classifiers for drifting concepts, *Intelligent Data Analysis* **11**(1): 3–28.

Stanley, K. (2001). Learning concept drift with a committee of decision trees, *Technical Report, Computer Science Department, University of Texas-Austin* .

Street, W. N. and Kim, Y. S. (2001). A streaming ensemble algorithm (sea) for large-scale classification, *7th ACM SIGKDD international conference on Knowledge discovery and data mining*, p. 377382.

Tsymbal, A. (2004). The problem of concept drift: definitions and related work, *Technical report*.

Vapnik, V. (1998). Structure of statistical learning theory, *Computational Learning and Probabilistic Reasoning* p. 3.

Wang, H., Fan, W., Yu, P. S. and Han, J. (2003). Mining concept-drifting data streams using ensemble classifiers, *9th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD 03, ACM, p. 226235. ACM ID: 956778.

Wang, Y. and Witten, I. (1996). Induction of model trees for predicting continuous classes, *Poster articles of the 9th European Conference on Machine Learning*.

Webb, G. I., Pazzani, M. J. and Billsus, D. (2001). Machine learning for user modeling, *User Modeling and User-Adapted Interaction* **11**(1): 19–29.

Widmer, G. and Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts, *Machine learning* **23**(1): 69–101.

Widyantoro, D., Ioerger, T. and Yen, J. (1999). An adaptive algorithm for learning changes in user interests, *Proceedings of the eighth international conference on Information and knowledge management*, ACM New York, NY, USA, pp. 405–412.

Zhang, P., Gao, B. J., Zhu, X. and Guo, L. (2011). Enabling fast lazy learning for data streams, *Proceedings of ICDM 2011*.

Zliobaite, I. (2010). Change with delayed labeling: when is it detectable?, *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on*, IEEE, pp. 843–850.

Author Biographies

George Giannakopoulos a research fellow at the National Center for Scientific Research "Demokritos" in Athens, Greece. He received his PhD degree from the University of the Aegean, Greece. In the past he has worked in various EU and national projects (e.g., OKKAM, OntoSum, SYNC3, CASAM). He has received one Best Paper award. He was the main organizer of the TAC2011 Multiling Pilot, has served as a program committee member and editorial board member for a variety of Artificial Intelligence, Machine Learning, Data mining and Natural Language Processing conferences and journals. His research interests cover machine learning and data mining, natural language processing, bioinformatics and the Semantic Web.

Themis Palpanas is a professor of computer science at the University of Trento, Italy. He received his PhD degree from the University of Toronto, Canada. Before joining the University of Trento, he worked at the IBM T.J. Watson Research Center. He has also worked for the University of California, Riverside, and visited Microsoft Research and the IBM Almaden Research Center. He is the author of five US patents, three of which are part of commercial products. He has received three Best Paper awards. He is serving on the Editorial Advisory Board of the Information Systems Journal and as an Associate Editor in the Journal of Intelligent Data Analysis. He is General Chair for VLDB 2013, has served on the program committees of several top database and data mining conferences.

Correspondence and offprint requests to: George Giannakopoulos, Institute of Informatics and Telecommunications, NCSR Demokritos, 15310 Aghia Paraskevi, Greece. Email: ggianna@iit.demokritos.gr